Meta-standardisation of Interoperability Protocols

By: Jorgina Kaumbe do Rosário Paihama
Supervisor: Associate Professor Hussein Suleman

Thesis Presented for the Degree of Master of Science
In the Department of Computer Science
Faculty of Science
University of Cape Town

June 6, 2012

Plagiarism Declaration

1. Plagiarism is to use someone else's work and pretend that it is ones own. I know that plagiarism is wrong.
2. Except for the parts acknowledged by referencing, this thesis is my own work.

Signed: Signed by candidate

Date: 06-June-2012

Acknowledgements

First and foremost I say "Thank you God for allowing me do this work, and for guiding & protecting me, always".

I would like to thank my parents Kundi Paihama and Maria do Rosário. You are everything to me, thank you for being the best and most supportive parents ever, I love you.

There are not enough words to express my gratitude to my supervisor Hussein Suleman, thank you, thank you and thank you again.

Last but certainly not least I would like to thank my big and awesome family, and all my wonderful friends. A Special thanks to my lab mates, you guys are the best.

Abstract

The current medley of interoperability protocols is potentially problematic. Each protocol is designed by a different group, each provides a single service, and has its own syntax and vocabulary. Popular protocols such as RSS are designed with simple and easy to understand documentation, which is a key factor for the high adoption levels. But the majority of protocols are complex, making them relatively difficult for programmers to understand and implement.

This research proposes a possible new direction for high-level interoperability protocols design.

The High-level Interoperability Protocol - Common Framework (HIP-CF) is designed and evaluated as a proof of concept that if interoperability is made simpler, then it can increase adoption levels, making it easier for programmers to understand and implement protocols, therefore leading to more interoperable systems.

HIP-CF is not suggested as the alternative to current production protocols. Rather it is suggested that the design approach taken by HIP-CF can be applied to other protocols, and also that a suite of simpler protocols is a better solution than various simple individual protocols.

Evaluation results show that current protocols can be substantially improved on. These improvements could and maybe should be the result of a deeper analysis of the goals of today's protocols and also a collaboration amongst the different groups that design high-level interoperability protocols.

This research presents a new approach and suggests future experimental research options for the field of high-level interoperability protocol design.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The amount of data available online increases on a daily basis[1] and the digital libraries community works to preserve that data and allow users to access it as easily as possible. Access to online information can be enhanced by allowing heterogeneous data providers to interoperate (communicate and exchange data amongst themselves).
This kind of interoperability is essential in many communities; an example is the benefits that interoperability among medical record systems would have to a doctor treating a patient for the first time. Instead of relying just on the patient's memory, the doctor could have access to the patient's complete medical history, thus being able to provide a more accurate course of treatment.
Currently there are many protocols that facilitate interoperability between systems at various levels of communication. Some are simple and can be easily implemented but may lack some efficiency while others are very efficient and just as complex [13]. Nevertheless they have been able to provide interoperability for Web resources.

1.1 Problem Statement

The current set of protocols are each created by a different group/community, they each provide a single main service, and they all have different syntax and semantics and therefore work differently. That may create implementation difficulties for programmers, who in most cases have to read long and complex documentations. These can be contributing factors to why some protocols have lower adoption rates than others.
Maybe interoperability and adoption rates can be improved if the protocols are simplified. Based on the number of users/adopters, it is quite clear that the Really Simple Syndication (RSS) is the most popular protocol [7]. The high popularity of RSS seems to be linked to the simplicity of the protocol and the easy implementation of an RSS feed. The specifications are presented in short and simple documents, that should not take a capable programmer more than a few hours to implement.
Looking at the RSS model it is easy to assume that it may be possible to improve interoperability by simplifying the protocols. To that end would it not be even better to simplify all high-level interoperability protocols and have one simple suite of protocols? By adding consistency, it is assumed that if it is simple to implement one particular

[1] http://www.internetworldstats.com/emarketing.htm

protocol, it may be even simpler to implement the next one with minimal incremental work. The possibility of improvement comes from the premise that if protocols had more in common it would be easier to provide solutions to interoperability problems.
Another problem with complex protocols is that some academic institutions, and even private and non-governmental organisations in developing countries may struggle to afford the necessary financial and skilled human resources required for the implementation and maintenance/support of such protocols. This increases the digital divide, and creates a even bigger gap that leaves behind those who could potentially benefit the most from greater interoperability. In the developing world interoperability has the potential to indirectly help solve many socio-economic problems, for example illiteracy, by giving people a chance at a better education through access to the latest international research output [11]. One example of a system that can be used for possible improvements on the level and quality of education is the Greenstone Digital Library. Greenstone as it is commonly known is a comprehensive system for the construction and presentation of information collections [99]. This system allows for collections to be presented over the Web or in a CD-ROM. Users can in a easy and mostly automated manner create and maintain collections of resources that can be accessed by browsing and/or searching. Greenstone caters for different document and metadata formats and allows even more to be supported with the development of plugins for new formats. By supporting the creation and maintenance of collections of resources, the ability to import new documents, create plugins, automatically create browsing and searching structures from the documents and other features that reduce and simplify the amount and complexity of work done by system administrators Greenstone could potentially be the starting point for institutions and individuals looking to maintain their digital data collections regardless of their limited resources.

1.1.1 Simplicity Success Stories

Some projects have managed to grow and survive even in very competitive times. Some of the success of these projects can be attributed to their simplistic approach.

1.1.1.1 Google

Google's success is attributed to a number of factors; the most relevant ones here are a simplistic approach to user interface design, innovative business model and focus on providing the best possible user experience.
Google started as a search engine and, they had to compete for market advantage with well established brands at the time, such as Altavista. Google's technological innovations, simple interface, and satisfying user experience made it a strong market competitor and ultimately the number one search engine in the world [21]. Altavista lost popularity at around the time Google started gaining market advantage, and this was attributed to amongst other things mismanagement and portal related clutter [96].
Another aspect of the Google model that is relevant for this research is how Google has created a suite of services. They have managed to create a one stop option for multiple needs by including services for emails, maps, translation, images, etc. And although some

services may be discontinued (e.g. Google Wave was shut down in April 2012[2]) the idea and concept of a suite of services has been working well for Google.
A suite of services as opposed to individual services is part of the approach proposed here.

1.1.1.2 Project Gutenberg

Project Gutenberg is a volunteer project that creates electronic versions of literary material to be freely distributed worldwide [70]. It is one of the oldest digital libraries [58], and the first to provide free electronic books (ebooks) [69].
Like the framework proposed here, and Google's, Project Gutenberg also provides a combination of different services. In addition to creating electronic versions of books, they also run projects such as [69]: the distributed proofreading project, where volunteers proofread new ebooks [58]; The sheet music project, where volunteers digitise public domain sheet music, which is periodically distributed for personal use and at libraries and schools by the CD and DVD project; and there is also the free kindle books project, that gives Kindle users access to most of the books in the Gutenberg collection.
Project Gutenberg works because it is a community driven project that requires minimal external funding [86]. It is a simple and consistent information digitisation project.

1.2 Proposed Solution

This research proposes the design of an experimental protocol that combines simplicity and efficiency to improve on interoperability, by combining various high-level interoperability services into one single suite of protocols that supports only the minimal required functionality without compromising efficiency.
The proposed solution is based on the hypotheses that if protocols were simpler and had more in common, it would be easier solve interoperability problems.
It is important to stress that this research does not suggest that protocol users stop using the current set of protocols to adopt the suite proposed here. This is merely an attempt to show that there is room for improvement in the current set of protocols and suggest a possible alternative direction for high-level interoperability research.
While the proposed suite will combine multiple high-level interoperability services, all with a common and consistent base protocol, each of the services is independent of each other, i.e. the implementation of one service should facilitate the implementation of another service given the acquired common knowledge, but it is not required that a user who implements one service must also implement the others.

1.3 Research Question

The goal of this study is to re-engineer the design of the current interoperability protocols and develop a set of specifications that can potentially improve interoperability by simplifying communication between data providers. The research question for this study is:

[2]support.google.com/wave/bin/answer.py?hl=en&answer=1083134

Is it possible to develop a uniform suite of simple and efficient interoperability protocols to improve on the current medley of protocols?

1.4 Scope and Limitations

This is purely an experimental research. The specifications produced by this research are not to be used as a protocol, but only as the basis of what high-level interoperability protocols can be.
The methodology used to complete this research work was:
Stage 1: An analysis of available research output in the areas of interoperability in general, and high-level interoperability protocols in particular. The latter involving everything from background, design and use to experiments of existing protocols. This stage also involved writing a research proposal with amongst other things, a research question that sets the scope of the research.
Stage 2: Gathering of primary data directly from the users. The research method used for the data collection was a survey, and the data was used as a form of user requirements for the design.
Stage 3: Protocol design. This involved creating the specifications for the base protocol and all services supported; deciding which features were crucial for simple and efficient services to work; and which features could be left out without making the protocol unusable.
Stage 4: Evaluation. This stage involved different ways of testing the protocol designed, and proving or disproving the research question.
Stage 5: Writing the thesis. This stage starts with stage 1 and continues until after all other stages are over, going through a series of drafts until the final version is submitted.

Thesis Outline

The remainder of this thesis is structured as follows:
Chapter 2 discusses the background and current state of different interoperability protocols;
Chapter 3 presents the result of a user survey that led to the design decisions of the experimental protocol;
Chapter 4 presents the experimental protocol suite;
Chapter 5 presents the different ways in which the evaluations were conducted;
Chapter 6 presents the conclusions drawn from the whole research process; and proposes possible ideas for the future of interoperability protocols research.

Chapter 2

Background

2.1 Interoperability?

Interoperability has many definitions across a wide spectrum of research areas. In computer science each of them is adapted to suit the context within which it is defined. They are similar because they all portray the same message of communication between heterogeneous systems.
In the ISO/IEC 2382-01 information technology vocabulary, interoperability is defined as "the capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units[1]". Interoperability also is defined as "the capability of different programs to exchange data via a common set of exchange formats, to read and write the same file formats, and to use the same protocols[2]". This research adopts a definition very similar to the latter. Here interoperability is defined as: the capability of different systems to communicate and exchange data with each other, using a set of predefined formats and protocols that will allow the systems to use each other's services successfully.
Successful communication and exchange of data and/or other resources is obtainable at various levels of a computer system, therefore there are different types of interoperability that can be achieved at different levels of abstraction, namely:
Syntactic interoperability allows systems to exchanging data using pre-defined data formats and common communication protocols[3], in other words using the same syntax. Syntactic interoperability is a requirement for any other level of interoperability to be achieved.
Semantic interoperability is the ability to accurately interpret the information exchanged and produce results as expected and understood by both parties [67].
Pragmatic/functional interoperability relies on a common set of functional primitives or on a common set of service definitions [28].
Technical/basic interoperability is obtained from using common tools, interfaces and infrastructure providing the user with a universal API [28].

[1]ISO/IEC 2382-01, Information Technology Vocabulary. Fundamental Terms
[2]http://en.wikipedia.org/wiki/Interoperability
[3]hl7.org

Legal interoperability is concerned with the legal implications of free access to digital information and intellectual propriety rights and ownership issues [52].

2.1.1 The Importance of Interoperability

A great deal of research work is carried out for the creation and preservation of digital information systems. The benefits of such work could be even more if all those different systems could work together in the preservation of information. This level of interoperability in digital library (DL) systems could increase information accessibility, promote open access, allow easier creation of federated metadata archives, improve efficiency and reduce costs [44]. Financial costs would be reduced by saving on development costs as well as storage costs as fewer repeated records may be necessary (because if one system has a record that a user needs, the user can get that by accessing that system via any another system).

2.1.2 Interoperability Protocols

A protocol is a set of formal rules that determine the way in which two systems communicate [68]. Protocols can be either low-level protocols or high-level protocols.
Low-level protocols define the physical and electrical characteristics of the communication [68].
High-level protocols define the data formats for message encoding and information control, message syntax, syntax for communication between devices, flow control and error handling [68]. The experimental protocol framework proposed here is a high-level interoperability protocol that works at the application layer of the Open Systems Interconnection (OSI) network model [66].

2.2 Protocol Message Exchange

Protocols allow systems to communicate by sending each other messages. These messages can either be a request for something or a reply to a request. Since the messages are exchanged between systems they need to be machine readable and encoded in a way that both parties understand.
High-level Web based protocols are either SOAP or REST based protocols.
SOAP is an XML message-based protocol for exchanging structured information in the implementation of Web services regardless of the operating system and implementation environment of the systems involved [65]. SOAP relies on XML for message formatting and on other Application Layer protocols for message negotiation and transmissions, usually Remote Procedure Call (RPC) for negotiation and Hypertext Transfer Protocol (HTTP) [24] for transmissions. The SOAP architecture contains different layers of specifications for: message formats, message exchange patterns (MEP), underlying transport protocol bindings, message processing models and protocol extensibility.
REST or Representational State Transfer is an XML-based architectural style that invokes Web services over HTTP by focusing on the roles of the architectural components of the Web, namely the origin servers, gateways, proxies and clients (the constraints faced by components to interact with each other and their interpretation of crucial data

elements) while ignoring the details of the components' implementation and syntax [25]. The term REST was introduced in 2000 by Roy Fielding in his Doctor of Philosophy Dissertation [25]. REST is an architecture and not a protocol. REST architectures consist of a client that makes a request to a server that returns an appropriate response. This communication is built around the concept of transferring representations of resources. A representation of a resource is a document that describes the current state of a resource. A REST client can be in one of two states[4]:

1. Transitioning between application states: a client starts sending requests when it is ready to transition to a new state, and is considered to be transitioning states when one or more requests are outstanding;

2. At rest: in this state the client is able to interact with its users but cannot create a load and does not consume any per-client storage on the servers or on the network.

The World Wide Web (WWW) implementation is an example of a system, possibly the largest system, that conforms to the REST specification[2]. Some of the existing high level protocols based on REST are: RSS, Atom, APP, OAI-PMH and OAI-ORE.
According to the MedBiquitous paper "Knowing When to REST: Simple Object Access Protocol vs. Representational State Transfer Web Services" [78], SOAP is better suited for activity oriented services such as transferring funds or booking a flight ticket and REST is better suited for resource oriented services that allow different operations to be applied to a dataset or object. A resource is information that has an identifier (URL) and a representation (a Web page).

Some of the most notable interoperability protocols are discussed below: for syndication RSS and Atom, for search Z39.50 and SRU, for harvesting OAI-PMH, for editing and publishing as well as description and exchange APP and OAI-ORE and SWORD for deposit.

2.3 Really Simple Syndication

Really Simple Syndication, commonly known as RSS, is an XML-based Web content syndication standard that periodically checks Web sites in search of updated content that is then delivered to subscribers (mobile and/or desktop devices) through an RSS feed [74][101].

2.3.1 Syndication History

The development of RSS involved the work of some of the Web syndication pioneers. From March 1999 to March 2009 RSS has had a total of sixteen published versions. The first versions of RSS, then known as RDF Site Summary, was published by Netscape on the 15th of March 1999. The author was Ramanathan Guha, one of the co-authors of MCF, and was called RSS 0.90 [72]. Just 3 months later, Netscape published a changed version as RSS 0.91, designed by Dan Libby, in this this version the RDF elements of were

[4]http://en.wikipedia.org/wiki/Representational_state_transfer

replaced by ScriptingNews syndication format and it was renamed to Rich Site Summary. The third version, still called RSS 0.91, was published in July 2000. This version was designed by Dave Winer, responsible for scriptingNews and published by UserLand, who also published RSS 0.92 in 2000 and later RSS 2.0 in 2002. It was in RSS 0.92 that the enclosure element was introduced, that allowed RSS feeds to carry audio files and was an important tool in the development of podcasts [98].
In December 2000 the RSS-DEV Working group released RSS 1.0 a version that introduced a number of changes such as: the name RDF Site Summary was reclaimed as support for RDF was reintroduced, support for XML namespaces, and adoption of the Dublin Core metadata format [98].
RSS 2.0 version 2.0.1, was published in 2003 by the Harvard Law School and once again was authored by Dave Winer and that is when the name Really Simple Syndication was formalised. All the subsequent versions from July 2003 to March 2009 were published by the RSS Advisory Board[5] , a group formed in July 2003 to maintain RSS 2.0. The group published the versions ranging from RSS 2.0.1-rv-1 to the current version 2.0.11, which was published on the 30 of March 2009.
Each successive version of the protocol aimed at improving issues in its predecessors, but that was not always an easy task as different versions were published by different groups (as they "battled" for official claim of the RSS format) and sometimes that meant making changes that were not necessarily improvements, for example the back and forth introduction and removal of RDF elements mentioned above. In terms of interoperability one known problem with RSS is that although there is no limit to the number of items in a feed, some news aggregators do not support RSS files larger than 150KB and therefore it is better to keep the files size under 150KB [98].

2.3.2 How RSS works

RSS provides subscribers with an organised list of notifications about new and updated Web content [82]. These lists are called RSS channels or feeds. A channel consists of a number of entries, which can be news headlines, full-text articles, article excerpts, weather reports, podcasts, etc. Each contains a set of metadata elements e.g. title, link, description [75]. A feed can be accessed by anyone who is interested in the content it provides. Maintaining an RSS feed involves creating an RSS document (an XML encoded file) that is Web accessible to RSS aggregators.
An RSS aggregation or RSS reader is a computer program that on a regular basis automatically accesses RSS feeds in search of updates, which are then presented to subscribers with the most recent entries at the top of the list [82]. If an item that is of interest to a subscriber is just a summary of an entry, the subscriber can follow the link to access the complete entry. There is a wide range of aggregators available. Some aggregators are accessed through a browser while others run as stand-alone applications on the subscribers devices.

[5]http://en.wikipedia.org/wiki/RSS

2.3.3 RSS Document

An RSS document is an XML file that conforms to the RSS protocol specifications and XML 1.0 specifications as recommended by the World Wide Web Consortium (W3C [95]) [74], see figure 2.1. The document starts with an XML declaration followed by the <rss> element, which contains the mandatory attribute version [74]. A more recent version of RSS does not invalidate the previous ones; therefore the attribute version is mandatory because it specifies which version of RSS the document conforms to. For example, if a document conforms to the current version of RSS, the attribute version will have a value of 2.0.
Next is the <channel> element. A document only can have one channel, which contains the information about the channel (metadata) and its contents [74]. The channel element has a set of mandatory and optional sub-elements [75] (see Appendix A).
A channel contains one or more items. An <item> often represents a story, and can have one or more of the optional sub-elements shown on table 2.1.
An item can be a complete story, whereby the link element is omitted, or a partial story, which has a link to the resource that tells the complete story. Although all the elements of item are optional, an item should always have either a title or a description.

Element	Description
title	The title of the item.
link	The URL that links to the item.
description	A summary of the item's content.
author	The email address of the author of the item.
category	One or more categories that the item belongs to.
comments	The URL of a page for comments made about the item.
enclosure	The description of a media object that is attached to the item
guid	A unique identifier of the item.
pubDate	The date at which the item was published.
source	The RSS channel that the item came from

Table 2.1: RSS Item Elements

An RSS feed file can either be manually created using any text editor or automatically created using one of the many programs available (e.g. Blogger or Radio) [82].

2.3.4 Validating a Feed

The validity of an RSS document can be tested using the RSS Validator [74]. All that is necessary to validate a document is to submit the RSS feed URL (Uniform Resource Locator) to the feed validator [72].The validator sends a response that tells the programmer that the document is valid or, if it is not, it specifies the errors and gives instructions on how to correct them. The validator also brings up warnings for elements that are not necessarily wrong, but that can be changed to improve the quality of the feed.
The complete and valid RSS (XML encoded) file may be placed on a Web site like any other Web page, and it is then accessible to any RSS aggregator/reader [82].

```
<rss version="2.0">
<channel>
<description>Masters Thesis</description>
<link>http://uct.example.com</link>
<title>RSS Document Masters Thesis Example </title>
<category>Academic</category>
<category domain="dmoz">Digital_Libraries_Laboratory/Computer_Science_Department/
University_of_Cape_Town/South_Africa</category>
<cloud domain="server.example.com" path="/rpc" port="80" protocol="xml-rpc" reg-
isterProcedure="cloud.notify" />
<copyright>Copyright 2009 UCT</copyright>
<docs>http://www.rssboard.org/rss-specification</docs>
<generator>RSS Playground 1.0</generator>
<image> <link>http://image.example.com</link>
<title>RSS Document Masters Thesis Example</title>
<url>http://www.uct.ac.za/images/uct.ac.za/about/logo/logocircless.jpg</url>
<description>AN example of the image element for a RSS document</description>
<height>32</height><width>96</width></image>
<language>en</language>
<lastBuildDate>Mon, 03 Aug 2009 18:33:44 GMT</lastBuildDate>
<managingEditor>JPaihama@example.com (Jorgina Paihama)</managingEditor>
<pubDate>Mon, 03 Aug 2009 22:00:00 GMT</pubDate>
<rating>(PICS-1.1 "http://www.rsac.org/ratingsv01.html" l by "webmaster@example.com"
on "2006.01.29T10:09-0800" r (n 0 s 0 v 0 l 0))</rating>
<skipDays><day>Sunday</day></skipDays>
<skipHours><hour>13</hour><hour>23</hour></skipHours>
<textInput>
<description>RSS documents are very simple to create.</description>
<link>http://www.uct.ac.za/textinput.php</link>
<name>query</name>
<title>TextInput Inquiry</title>
</textInput> <ttl>60</ttl>
<webMaster>JPaihama@example.com (Jorgina Paihama)</webMaster>
<item>
<title>Meta-standardisation of Interoperability Protocols Proposal</title>
<link>http://www.cs.uct.ac.za/2009/05/12/proposals.htm</link>
<description>Access to online information can be enhanced by allowing
heterogeneous data providers to interoperate.</description>
<guid>http://www.cs.uct.ac.za/2009/05/20/proposals.htm</guid>
</item>
<item>
<description>Prototypes</description>
<guid isPermaLink="false">2009-05-06+lifestyle+joebob+1</guid>
</item>
</channel>
</rss>
```

Figure 2.1: RSS Document

2.3.5 RSS Autodiscovery

RSS autodiscovery allows Web browsers and other software programs to automatically find a website's RSS feed [73]. This feature is supported by popular browsers like Mozilla Firefox 2.0 and Microsoft Internet Explorer 7.0 (and all subsequent versions). When a browser opens a page that provides a syndication feed, it displays the RSS feed icon in the address bar. The users can click on the icon to access and/or subscribe to that specific feed. Autodiscovery is not an automatic feature; it is optional and has to be enabled by the feed programmer by including in the server implementation the <link> element, for example <link rel "alternate" type="application/rss+xml" title="SomeTitle" href="http://www.example.com">.
Websites that do not have autodiscovery enabled can indicate their support of RSS feeds by having an RSS, XML or a 'Syndicate This' link on their pages.

2.3.6 Latest issues

In August 2008 Netscape finalised the transition of the first two versions of RSS (RSS 0.90 and RSS 0.91) to the RSS Advisory Board. All the specification documents, DTDs (Document Type Definitions) and help files are now hosted by RSSBoard.org which provides guidelines for a smooth transition for all users. There are constant changes made to the RSS specifications [74], and the latest proposed change, which is still under discussion and may be considered for approval on the basis of the arguments presented, is adding an RSS namespace. Adding a namespace to RSS would allow RSS elements to be embedded inside XML elements that have a default namespace. RSS simplicity is a controversial issue. While some people say that it is excessively simple and therefore lacks structure/semantics that are crucial for improved levels of functionality and security [33], the vast majority seems to see simplicity as its main success factor [100], and that can be assumed to be true considering that a large number of Web sites have implemented RSS feeds. Those include sites for major news broadcasters like CNN, BBC and the New York Times [82]. RSS is seen by some as being synonym to syndication [100].

2.4 The Atom Syndication Format

The Atom Syndication Format (also called Atom 1.0 or just Atom) is a syndication format that represents Web content and metadata [59]. It provides a simple way to read, write and publish Web resources of regularly updated websites. This protocol is part of the work developed by the Internet Engineering Task Force (IETF) Atom Publishing Format and Protocol Working Group (known as AtomPub). Created around July 2005, the main goal for Atom was to upgrade RSS in providing syndication of Web resources (e.g. weblogs and news headlines) to Web sites or even directly to users.
An Atom document is an XML document identified with the application/atom+xml media type, which provides lists of related information known as feeds [59] (see Listing 1). A feed in Atom is equivalent to an RSS feed, but obviously with differences in the elements and guidelines. There are two types of documents in Atom: Atom Feed Documents and Atom Entry Documents. An Atom Feed Document represents an Atom feed and therefore includes a complete or partial list of all entries and their corresponding

metadata. An Atom Entry Document contains only one Atom entry, and is characterised outside of feed context.

```
<?xml version=1.0 encoding= utf-8?>
<feed xmlns=http://www.w3.org/2005/Atom>
<title> Example Atom feed</title>
<subtitle> Feeds subtitle</subtitle>
<link hreflang=en href=http://example.org/feed/ rel=self />
<link href=http://example.org/ />
<updated> 2009-08-07-T15:43:35Z</updated>
<author>
    <name>Jorgina Kaumbe do Rosario Paihama</name>
    <email>jpaihama@example.org</email>
    <uri>http://www.my_domain_example.org</uri>
 </author>
<id>urn:uuid:60a76c80-d399-11d9-b91c-0003939e0af6</id>
<entry>
    <title>Example Title</title>
    <category term=Example Category/>
    <link>href=http://example.org/2009/08/07/atom03</link>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <published >2009-08-07-T15:43:35Z </published>
    <updated>2009-08-07-T15:43:35Z</updated>
    <summary>A short summary of the item.</summary>
    <content> The resource content</content>
</entry>
</feed>
```

Listing 1: XML encoded Atom document

2.4.1 Constructs

Atom supports three kinds of constructs: the text construct, the person construct and the date construct [59]. An element inherits all the requirements from its construct definition, such that a text element inherits all the requirements of a text construct [59].

2.4.1.1 Text constructs

A text construct is an element that contains language-sensitive human-readable text [59]. A text construct "type" attribute may have the value of: Text, HTML or XHTML.

2.4.1.2 Person constructs

A person construct is an element that describes a person, a corporation or a similar entity [59].

2.4.1.3 Date constructs

A date construct [59] is an element whose contents conform to the "date-time" specification in RFC 3339[38].

2.4.2 Atom Elements

There are two types of elements in Atom: the container elements and the metadata elements [59]. Both types of elements are described in Appendix B.

2.4.3 How Atom differs from RSS

Atom advocates have pointed out various reasons why Atom differs from RSS and why it is better. Some of these reasons are:

- Atom provides a way to clearly label the type of content provided by each individual entry (RSS does not), and in addition to plain text and escaped HTML (both supported by RSS), Atom supports XHTML, XML, Base64-encoded binary as well as references to external media content (e.g. docs, audio and video streams) [76].
- Instead of the RFC 822 [15] formatted timestamps used in RSS, Atom's created and updated elements timestamps conform to the RFC 3339 [38] specification, which is a subset of ISO 8601, which represents dates and times using the Gregorian calendar [38].
- Atom uses the standardised xml:lang attribute which allows the specification of the language for each individual item in an entry as well as the feed language [76]. RSS only allows the specification of the feed's language, and assumes that each individual entry uses the same language.
- Atom supports the use of Internationalised Resource Identifiers, which allows links to resources and unique identifiers to use characters other than the ASCII character set [81].
- The Atom vocabulary can include markup reused from other vocabularies - in other words, it can be used outside the context of an Atom feed [59].
- Atom allows the distinction between a partial and a full resource in a feed by using <summary> and <content> elements respectively. RSS only has a <description> element and from that it is not possible to tell whether it contains the full resource or just an excerpt [76].
- Atom 1.0 is part of an XML namespace and can contain elements from other namespaces [76]. Except where explicitly forbidden, Atom allows foreign markup in any Atom document; all rules that apply to this are explained in RFC 4287 [59].

2.4.4 The Atom vs. RSS dispute

Atom was designed to be a substitute for RSS. The intention was to clarify RSS ambiguities, bring stability to syndication by consolidating its multiple versions as a registered official RFC document, provide extensibility for licensing, versioning and access control content areas [51] [3] [59]. Despite the fact that it has been adopted by many individual users and influential companies such as Google, Atom has not (in terms of popularity and use) surpassed RSS. Some of the reasons for Atom's inability to replace RSS are[6]:

- Usually sites that publish Atom also publish RSS, while the opposite is not always true.
- RSS support for enclosure led to the development of podcasting, and big companies like iTunes still have RSS as their main syndication format, although they also support Atom.
- Major news networks like CNN, BBC and the New York Times choose to publish their feeds only in one format, and their choice is RSS. However today many processing applications support both RSS and Atom.
- Loyal RSS users see no reason to use another format, given the fact that RSS provides all the functionality that they require.

The contents of an Atom document can be secured using XML security mechanisms, digital signatures or encryption. Atom also provides support for autodiscovery (with the IANA registered application/atom+xml MIME type) and a "self" pointer. The pointer allows news readers to auto-subscribe using Web-standard dispatching techniques [76]. In summary, Atom provides all the functionality of RSS with a more structured framework.

2.5 The Z39.50 Protocol

The Z39.50 protocol is a ANSI/NISO application layer protocol that supports distributed search and retrieval between structured network services [34]. This protocol stipulates data structures and interchange rules that allow a client machine to search and retrieve records from databases on a server machine, across different platforms [50]. This protocol is widely used by librarians, very often integrated into library systems and personal bibliographic reference programs (e.g. interlibrary catalogue search with Z39.50 queries). The Z39.50 protocol was first adopted in 1988, followed by an extensively revised version 2 in 1992 and version 3 in 1995 [49]. A revised copy of version 3 was published later in 2003 [4].

2.5.1 How does Z39.50 work

Communication between a client and a server is established via a Z39.50-Association, which is established by a client and can be terminated by either part (or implicitly by loss of connection) [4]. More than one Z39.50-Association can be established per connection.

[6] http://en.wikipedia.org/wiki/Atom_rss

The Z39.50 protocol supports search, retrieval, sort and browse functions. Here services are described as the processes carried out via message exchange by client and server [4]. A service can be in one of three states:
1. confirmed (a client request followed by a server response);
2. non-confirmed (a request from a client or server with no corresponding response);
3. conditionally-confirmed (a service that may be in either state 1 or state 2).
An operation involves all processes from an *initiating request* to a *terminating response*, including everything in-between, and can only be initiated by certain clients. Z39.50-2003 standards [4] supported nine types of operations, namely: Init, Search, Present, Delete, Scan, Sort, Resource-report, Extended-services and Duplicate Detection.
The protocol works as follows:

- The client and server establish a connection negotiate expectations and limitations on the activities that will occur (things like version used and maximum record size).
- After these agreements are negotiated, the client may submit a query. The request includes the query and the name of one or more databases to be searched.
- The server executes the search against a database(s), and a result set is created. A search can be executed using one of six attributes: completeness, position, relation, structure, truncation and use.
- The client asks for records from the result set or requests that the server performs some additional processing of the result set before sending it to the client.
- After receiving the response set the client machine may do some further processing (depending on the interface software) on the records before displaying them to the user.

2.5.2 Reported Issues

Z39.50 is an application layer protocol within the OSI reference model, and therefore supports lower level OSI services [49]. Despite that, many of the protocol implementers chose to layer Z39.50 on top of TCP/IP, as opposed to implementing it in an OSI environment to benefit from the full OSI services. The main reasons for this choice were:

1. The size and complexity of OSI implementation was daunting;
2. There was no mature OSI software available for the full range of computing environments in use at the implementing institutions;
3. The architectural structures within the OSI application layer were seen as unstable; and
4. Some implementers were concerned that the complexity of the OSI upper layers would outweigh whatever benefits this implementation had to offer.

The improvements presented by each consecutive version did not stop critics from slating Z39.50. Amongst other reasons, the protocol was criticised because it required a particular

type of software to be installed, configured and maintained in order to work, it was expensive, and even though there was search and retrieval, Z39.50 did not allow bulk data exchange. Due to this drawback, researchers hope that newer technologies like XML and RDF would fill the gaps left by the Z39.50 protocol.

2.5.3 Current State

The development of the Z39.50 protocol predates the launch of Web technologies, and it did not succeed for very long after that. Although still used by a few, today Z39.50 is considered by most to be an obsolete protocol. The fact that now there are many other protocols that provide an improved level of functionality has not stopped Z39.50 advocates from trying to bring the protocol back to life, and since 2002 they have been working towards this goal. The newly renovated ZING protocol (Z39.50 International: Next Generation) is an attempt to revive Z39.50 by adopting new standards like XML and SRU [46].

2.6 Search/Retrieval via URL (SRU)

Search/Retrieval is a service for search and retrieval of Web resources across the Internet. A client makes a search request for the retrieval of matching records from the server [47]. This standard is based on the Z39.50 protocol. The protocol can be used in two different ways [83]:

1. As parameters in a URL, called Search/Retrieval URL Services or SRU; or
2. SRU via HTTP SOAP, formerly know as Search/Retrieval Web Services or SRW [48].

This protocol has two published versions 1.1 and 1.2 and an OASIS draft for SRU 2.0.

SRW used a SOAP (Simple Object Access Protocol) interface and the Common/Contextual Query Language (CQL) to allow a search function that promotes interoperability between distributed databases by providing a common utilisation framework [83]. SRU (Search/Retrieval via URL) is a standard search protocol for Internet search. It employs a URL encoded HTTP interface and, like SRW, it uses CQL. CQL is a formal language for representing queries to information retrieval systems [47]. Known as Common Query Language in version 1.1, it was later changed to Contextual Query Language in version 1.2. This query language was designed to support queries that are human readable and writable while maintaining the efficiency of more expressive languages like SQL and XQuery [83]. The syntax for SRU is specific for both the queries and how the results are presented [54].
SRU is XML based and supports three main functions [83]:

1. Search

The most important function of these protocols is search/retrieve. This is done using the SearchRetrieve operation, in which the client sends a SearchRetrieveRequest with the

necessary parameters and the server responds with a SearchRetrieveResponse that is a list of XML records and the full count of the number of records that matched the query.

2. Browsing (done via the scan request)

SRU also provides the Scan request, which only differs from search in that it has only one search clause that specifies the index relation and term. This allows the user to browse through the information in the records one by one. The scan request returns a portion from the sorted list of terms in the database for a given index.

3. Server capability

The final operation provided by this protocol is a server capability request that allows the client to find out which protocols the server supports and what it supports in terms of CQL. The messages exchanged here are known as the Explain request and response.

2.6.1 Search Requests and Parameters

SRU allows the user to send at least 6 different combinations of search URL requests[7]. The optional parameters can be mixed and matched to meet the users needs. Below are examples of requests sent via the URL, and Table 2.2 provides descriptions for each individual parameter.

1. Explain Request: http://z3950.loc.gov:7090/voyager
2. Simple term search: http://z3950.loc.gov:7090/voyager?version=1.1&operation =searchRetrieve&query=dinosaur
3. Simple term search to retrieve a specified number of records: http://z3950.loc. gov:7090/voyager?version=1.1&operation=searchRetrieve&query=dinosaur& maximumRecords=1
4. Simple term search to retrieve a specified number of records in a specified metadata format: http://z3950.loc.gov:7090/voyager? version=1.1&operation searchRetrieve& query=dinosaur&maximumRecords=1&recordSchema=dc
5. Simple term search to retrieve a specified number of records, starting from a specified position in the storage medium, displaying a specified number to records in a specified metadata format. http://z3950.loc.gov:7090/voyager? version=1.1&operation=searchRetrieve&query=dinosaur&startRecord=2 &maximumRecords=5&recordSchema=dc
6. Term search at a specific place within the text, for example search for the term name in the abstract, or the term learned in the conclusion. http:// z3950.loc.gov:7090/voyager?version=1.1&operation=searchRetrieve&query= title=dinosaur

[7]http://www.loc.gov/standards/sru/simple.html

Parameters	Description
version	Specifies the version of the protocol used
operation	Specifies the type of service requested
query	Term(s) used to find records of interest available in the server
maximumRecords	Specifies the maximum number of matching records to be returned for the request
recordSchema	Specify the metadata format of the records in the response
startRecord	Requests that, when searching through its records, the server only returns matching records that appear after a specific position/number on the records index

Table 2.2: SRU Parameters and Descriptions

2.7 The Open Archives Initiative –Protocol for Metadata Harvesting (OAI–PMH)

2.7.1 The Open Archives Initiative (OAI)

The Open Archives Initiative allows Web-accessible repositories to interoperate by sharing, publishing and archiving one another's metadata records [60]. The OAI started within the e-prints community, supported by an increasing need for a low-barrier solution for interoperability among heterogeneous repositories. The expected benefits from this initiative were [60]:

1. To allow digital material to be accessed more widely and used for purposes other than the ones that led to its creation.

2. The assumption that the possibility of accessing multiple repositories will improve the types and quality of value-added services provided to users.

3. The potential to provide cost-effective means of communication for the academic community and the expectation that the rapid growth of digital material because of the Internet will increase the target market of many repository systems.

Initially focussed on improving access to e-print repositories, OAI now also provides support for other types of repositories or digital material [60]. Currently OAI maintains two projects: the Open Archives Initiative - Protocol for Metadata Harvesting (OAI-PMH) and the Open Archives Initiative - Object Reuse and Exchange (OAI-ORE).

2.7.2 Overview

OAI-PMH [60] is a protocol that provides an application-independent interoperability framework for metadata harvesting. The main objective to be achieved with the OAI-PMH was simplicity [55]. This protocol was designed to be easy to implement (based on widely accepted standards such as HTTP, XML and Dublin Core) and still highly efficient.

2.7.3 History

The first step towards OAI-PMH was the Santa Fe convention[92], organised by Paul Ginsparg, Rick Luce and Herbert Van de Sompel of the Los Alamos National Laboratory, in October 1999. A group of technical experts proposed the development of the Universal Preprint Service (UPS)[60][91]. The UPS was a universal service for author self-archived scholarly literature, and its target was the dissemination of e-prints (an e-print is an author self-archived document).The aim of the Santa Fe convention was to discuss interoperability issues, establish a forum for ongoing work on interoperability of self archiving solutions, and agree on the development on a prototype digital library service based on the main e-prints repositories. Due to the fact that not all e-prints were Preprints[8] and because the name UPS was already an established brand name for a delivery company, the name of the framework soon changed from UPS to OAI. From agreements reached at the convention (issues like: transport protocol, metadata format and quality assurance, intellectual properties and usage rights), the Santa Fe Convention protocol for metadata harvesting was created. Two other versions of the protocol were created after that. Table 2.3 shows the developments in the protocol from Santa Fe convention to the current OAI PMH version 2.0 [57].

	Versions		
	Santa Fe Convention	**OAI–PMH v.1.0/1.1**	**OAI–PMH v.2.0**
FEATURES			
Nature	Experimental	Experimental	Stable
Verbs	Dienst	OAI–PMH	OAI–PMH
Request Type	HTTP GET/POST	HTTP GET/POST	HTTP GET/POST
Response Format	XML	XML	XML
Transport	HTTP	HTTP	HTTP
Metadata	OAMS	Unqualifies Dublin Core	Unqualified Dublin Core
About (Content)	eprints	Document like objects	Resources
Model	Metadata Harvester	Metadata Harvester	Metadata Harvester

Table 2.3: OAI PMH Version History

There are two common options available for building interoperability services based on distributed metadata records: the cross archive searching approach and the harvesting approach [55].
The cross archive searching approach is based on a synchronous model in which a user's search query generates requests to all participating metadata archives. The search approach has the following characteristics [55]:

- Search queries can be directly encoded (e.g. contributor="Jorgina");
- Service providers do not need to have a central database for the metadata they collect;

[8]A Preprint also known as an E-print is an author's self-archived document derived from scientific or other scholarly type of research .[60]

- Service providers must solve duplication, ranking and merging problems simultaneously; and
- The service performance depends on the slowest data provider.

The harvesting approach on the other hand has the following characteristics [55]:

- The harvesting protocols have either a very basic search vocabulary or none at all;
- Service providers need to have a database in which to store the metadata records from the participating repositories;
- Using the harvesting approach leads to the existence of duplicate records;
- Service providers need to frequently harvest the participating repositories in order to provide the most recent information possible. For that the provider has to deal with updated and deleted records.
- Providing value-added services while solving problems such as duplication, ranking and merging is not time critical for service providers because all these processes can be conducted simultaneously.

As the name suggests, OAI–PMH uses the harvesting approach. This choice is based on the fact that the cross archive search approach does not provide the necessary scalability for large numbers of participating repositories [55]. The lack of scalability in the cross archive search approach arises from the fact that the service performance always depends on the slowest data provider. For example, to use a large cross archive search-based service the users have to first select the number of archives to be searched by the query submitted; the query will be processed by all participating archives and a response will be delivered only after all archives have processed the query. A parallel search is in most cases impossible and not at all recommended by the service providers, therefore delays are to be expected in this approach [55].
This protocol aimed to:

1. Surface hidden resources,
2. Provide access to the archive's metadata,
3. Provide low cost interoperability, and
4. Support a new pattern for scholarly communication by providing a world-wide consolidation of scholarly archives.

2.7.4 Framework

There are two actors in the OAI–PMH framework: a data provider and a service provider (see Figure 2.2).
A data provider uses OAI–PMH to expose metadata about repository content to service providers [42]. The data provider maintains one or more repositories.

A service provider uses OAI–PMH to harvest metadata from data providers. In the context of OAI–PMH, the term harvesting refers to collecting metadata from different repositories and the possible storage of all metadata into a central database. Value-added services are then provided to users based on the resources of this central database [42]. In most cases these resources are only metadata records and not the actual digital objects/content. So value-added services are a result of a combination of the harvesting approach and other mechanisms (these mechanisms are defined according to the service provided).

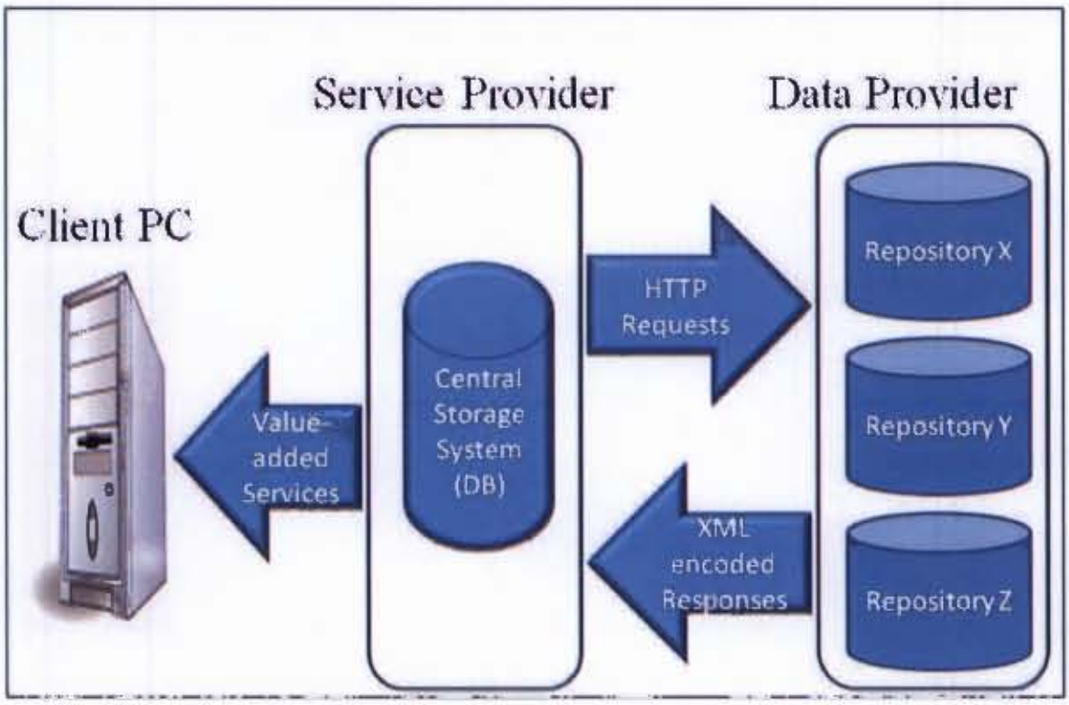

Figure 2.2: OAI-PMH flow of activities

OAI–PMH communication is based on HTTP requests from service providers and XML responses from data providers [55][24]. Since both HTTP and XML are well known standards, the deployment of this protocol tends to be flexible and it can be configured in any different ways as listed below [60].

1. **Multiple Service Providers:** multiple service providers harvesting metadata (directly) from multiple data providers.

2. **Aggregators:** service providers harvesting metadata from the data providers via an aggregator. The aggregator harvests the metadata from the data providers and this data is then collected by the service provider.

3. **Harvesting plus searching:** The combination of harvesting with some search based protocol, for example Z39.50 or SRW. Aggregators harvest the data from data providers and then service providers search the data aggregators.

2.7.5 Communication

The data and service providers communicate via the HTTP protocol. The service provider sends a GET or POST request to the data provider and in response gets an HTTP response with the text/xml content type for a successful request and HTTP or OAI error codes for when an error occurs.
There are six types of requests (also called verbs) that allow communication between the providers, namely: Identify, ListMetadataFormats, ListSets, ListIdentifiers, ListRecords and GetRecord [60]. While a service provider can choose which types of requests to support, data providers are required to implement all six.
Below is a brief description of how each verb works. Refer to the OAI–PMH documentation [45] for detailed explanation and examples.

2.7.5.1 Identify

The identify request retrieves information about the data provider/archive/repository. This verb has no required argument and will generate an error or exception when the request includes illegal arguments. A successful response will contain a repository name, base URL, version of the protocol, the earliest date stamp, information on support for deleted records, granularity support, administrator email address and, optionally, the compression encoding information and a description of the repository.

2.7.5.2 ListMetadataFormats

The ListMetadataFormats request retrieves a list of all metadata formats supported by a repository. This verb has only one (optional) argument: *identifier* (unique identifier of the specified item). The error and exception conditions identified with ListMetadataFormats are: badArgument (illegal/missing arguments), idDoesNotExist (the value of the identifier is unknown or illegal in the repository) and noMetadataFormats (no metadata formats available for the specified item).

2.7.5.3 ListSets

The ListSets request retrieves the set structure of a repository, used for selective harvesting. This verb may require the exclusive argument *resumptionToken*. A *resumptionToken* is a value returned with a previous incomplete response-this value is used to retrieve the remaining resources. An error may occur if the repository does not support a set hierarchy.

2.7.5.4 ListIdentifiers

The ListIdentifiers request retrieves a list of record headers. This verb has one mandatory argument: *metadataPrefix* (specifies metadata format); three optional arguments: *from* (datastamp lower bound for selective harvesting),*until* (datastamp upper bound for selective harvesting) and *set* (set criteria for selective harvesting); and one exclusive argument: *resumptionToken* . An error may occur if the combination of items that match the arguments from, until and set generates an empty list.

2.7.5.5 ListRecords

The ListRecords request retrieves records from a repository. This verb has one mandatory argument: *metadataPrefix*; one exclusive argument: *resumptionToken*; and three optional arguments: *from*, *until* and *set*.

2.7.5.6 GetRecord

The GetRecord request retrieves an individual metadata record from a repository. This verb has two required arguments: *identifier* (unique id) and *metadataPrefix* (specifies the metadata format of the document to be retrieved).

2.7.6 Metadata

OAI–PMH does not have a unique metadata format. The format used may be agreed upon by a community (any group of cooperating data and service providers), and should have the three following properties: 1. an id string to specify the format; 2. metadata schema URL; and 3. XML namespace URI. While individual communities are free to use any XML encoded metadata (e.g. MARC, SPECTRUM), the OAI–PMH framework mandates the use of unqualified Dublin Core to provide a basic level of interoperability [60].

2.7.6.1 Dublin Core (DC)

The Dublin Core is a 15 element metadata format. The elements are: *title, creator, subject, description, publisher, contributor, date, type, format, identifier, source, language, relation, coverage and rights.* The elements are all optional and all repeatable [39].

2.7.7 Advantages

- A lot of the features in the OAI–PMH are not mandatory; there is a minimal repository implementation list to be followed by implementers. The list[9] states the few features that must be implemented to be able to use the protocol without losing the main functionality. This is done to ensure the lowest possible barrier to metadata access.
- Multiple service providers harvesting from multiple data providers suggests wider dissemination of metadata [60].
- Acts as a under layer for building value added services [60].
- Supports any XML encoded metadata standard.
- Service providers can access all records or (based on set hierarchy and date stamps) specify the subset of records they need.

[9]http://www.openarchives.org/OAI/2.0/guidelines-repository.htm

2.7.8 Disadvantages/Limitations

- The OAI–PMH does not harvest the document itself; it only harvests the metadata record that relates to the documents available in a repository.
- OAI–PMH does not provide a search function for the aggregated data - all it does is bring the data together into a central location.

While the implementation of the protocol itself is said to be technically simple and there are even toolkits available for that [60], the challenge lies in building services that satisfy user needs.

2.8 The Atom Publishing Protocol (APP)

APP also known as AtomPub is an HTTP and XML [95] based application level protocol for editing and publishing Web resources for constantly updated websites [79]. Also created by the AtomPub group APP is used to generate and manage collections of Web resources represented by the Atom Syndication Format. Those Atom formatted representations describe the state and metadata of the resources. Because it is based on HTTP, APP uses the HTTP operations (GET, POST, PUT and DELETE) to disseminate instances of Atom feeds and Entry documents. Entry Documents are documents/resources/entries that are members of a collection and are represented as Atom Entry Documents [30]. An Atom document is specified in XML format but with a few minimal modifications to some Infoset terms. Infoset terms are terms that provide a consistent set of definitions used in specifications that use well-formed XML documents [94], (e.g. element information item is referred to as element only). The protocol supports: collections, services and editing of the Web resources [30].

- A collection is a set of resources, that can be retrieved partially or as a whole.
- Services provided are the discovery and description of collections.
- Editing involves creating, editing and deleting resources.

The core of the APP is the collection of Web resources [30]. A collection contains all the resources that are part of a specific group, and every collection has a URI (Uniform Resource Identifier) that uniquely distinguishes it from the other collections in a Workspace. A Workspace is a named group of collections [30].
Items in a collection are called Member Resources and they have their URIs listed in the collection they are in. Member resources can be either Entry Resources (represented as Atom Entry Documents) or Media Resources (images, documents, audio and video files basically any media format supported by the collection).
A collection is represented by an Atom Feed Document, which contains an unlimited number of entries, listing of the URIs and metadata of its Member Resources[30].

2.8.1 How does APP work

The initial step to take when using APP is to find out what collections are available and the types of resources that the collections contain [79]. Afterwards the user can then use any of the protocol's supported functions. To find out what collections are available the client sends a GET request to the service document URI on the server [30]. The server responds by sending a service document (XML format) that lists the URIs of all the collections available as well as the services supported by each collection. Depending on the details of the client request, the service document may provide other details, for example authentication credentials [79].
A service document describes the workspace(s), within which collections are grouped in a logical manner. Each collection in a workspace has at least a collection href element that provides the URI of the collection and an accept element that specifies the type of content stored in the collection e.g. images, entry. A GET request to a collection URI returns an Atom Feed Document with a complete or partial list containing the URIs of the collection Member resources. Whether the list is complete or partial depends on the request parameters.
After identifying the collections available the user can use the service(s) provided by the protocol. APP uses HTTP requests to perform its designated activities. Figure 2.1 shows the flow of activities and the type of HTTP request used to achieve the main APP functions as explained below [79].

2.8.1.1 Create a resource

Entry resource
To create a new Entry resource the user sends a HTTP POST request to a collection URI. Entries must have at least an id, an author and an updated element, but any of these client-given values may be overwritten by the APP server. The server responds by sending the status of the request (HTTP response status code, for example 200 for OK, 201 for created) and a location header containing the unique edit URI of the resource it has just created. In addition to that, if it has overwritten any of the client-given values (id, author and updated elements) the server can send a copy of the entry values that were stored in the collection.
Media resource
To create a Media resource the user issues an HTTP POST request to the collection URI, similar to creating an Entry resource. The difference here is that the user now sends the POST request and a representation of the media resource. If the collection supports the media format that the user wants to store, two resources are created: a Media Entry for the requested resource and a Media Link Entry for metadata about the resource. Media resources are also known as media-link entries [30].

2.8.1.2 Retrieve a resource

To retrieve a resource from a collection a user issues a GET request to the resource URI. The APP server responds by sending back a representation of the Member Resource.

2.8.1.3 Edit a resource

To edit an Entry resource the user retrieves the document from the collection using a GET request and the document URI. When the server sends the document the user can edit and then return the document to the collection using the PUT request. Editing a media resource is almost similar to editing an Entry resource. The user first issues a GET request to the URI of the edit-media link of the specific resource. This will retrieve an editable version of the resource, to which the user can make all the appropriate changes and then return the resource by issuing a PUT request to the edit-media URI.

2.8.1.4 Delete a resource

To delete an Entry resource from a collection the user issues a DELETE request to the appropriate edit URI. Deleting a Media resource is slightly different. A Media resource is deleted as a result of its corresponding Media Link Entry deletion.

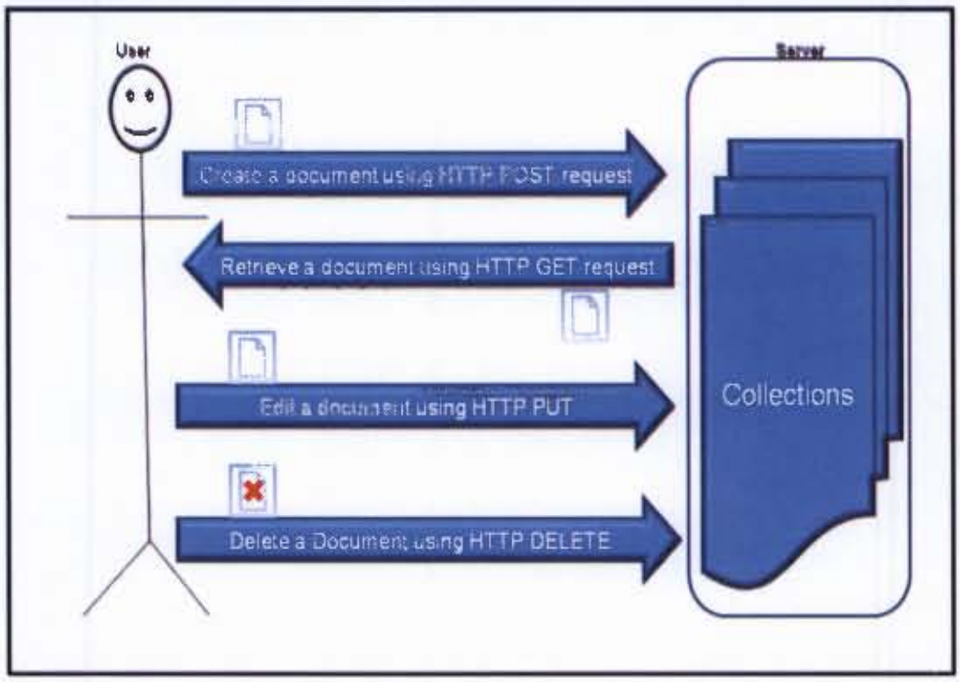

Figure 2.3: APP flow of activities

2.8.2 APP Security

Authentication to prevent unauthorised access is recommended but not required by APP [30]. The level of authentication, if any, is decided by the server administrator. Failure to provide some level of authentication exposes the APP implementations to risks such as denial-of-service, replay attacks, spoofing attacks or data loss [30]. Therefore it is recommended that servers provide at least support for HTTP authentication requirements [26] with TLS (Transport Layer Security) [71]. This basic level of authentication is sufficient

to establish a baseline for interoperability [30], but implementers can use alternatives that are at least as good or considered better than this minimalist approach.

2.8.3 Server Actions

APP imposes very few restrictions on server actions. Servers' behaviour will vary according to their individual configurations [30]. While APP specifies the use of GET, POST, PUT and DELETE requests, it does not imply that other HTTP requests cannot be used on collection resources. The same goes for HTTP status codes servers can choose to respond in any way (accept, reject, delay, censor or moderate a request), and client software must support all types of HTTP server response. A request can have a different HTTP response, or receive a different feed, or different entry contents on each individual server.
APP was initially designed to allow Weblog owners to upload media content into their posts, but because of its ability to provide support for a wide range of media resources it is now used for a number of different applications, such as document management, Office suites, photo libraries, podcasting, software distribution, video blogging, Wikis and XML repositories [80].

2.9 The Open Archives Initiative - Object Reuse and Exchange (OAI-ORE)

The Open Archives Initiative - Object Reuse and Exchange is a protocol that facilitates the description and exchange of Aggregations of Web Resources [43].

2.9.1 Aggregations

An Aggregation is a set of related resources that can be treated as a single resource, and a resource is defined as being any item of interest [43]. An example of an Aggregation is a website that contains multiple Web pages, each connected to the next and previous pages via hyperlinks. An Aggregation has the following characteristics [43]:

- Each consists of one or more resources, which can be stored on a single Web server or distributed across different Web locations.
- Resources in an aggregation can indicate their semantic relationship with other resources (e.g. resource X is a previous version of resource Y).
- Resources may be of a certain type (e.g. bibliography, table of contents).
- Aggregations and/or individual resources may have a relationship with other internal or external Aggregations/resources (e.g. citations).

Aggregations can be made visible to Web agents (i.e. browsers and crawlers) by assigning them machine readable identities/descriptions [43]. Those descriptions can be the basis of user interfaces that would allow users to navigate and manipulate the Aggregations.

Aggregations are described by Resource Maps. A Resource Map is an RDF graph that is serialised to a machine readable format, to provide a description about an Aggregation according to the OAI-ORE data model. The description includes details about the set of resources that the Aggregation contains and the existing relationship between an Aggregation and its resources. ORE provides guidelines for users to create and publish Resource Maps in different formats such as Atom, RDF/XML and RDFa.

2.9.2 Data Model

The OAI-ORE data model is built on the foundation of the WWW architecture, RDF, Cool URIs and Linked Data. This abstract data model includes the following four entities: Aggregation, Aggregated Resource, Resource Map (ReM) and a Proxy [43].

2.9.2.1 Aggregation

An Aggregation is a type *ore:Aggregation* resource that represents a set of other resources. This type *ore:Aggregation* is associated with a resource via an assertion by at least one Resource Map. An Aggregation is a conceptual construct and therefore does not have a representation, but the Resource Map that asserts the Aggregation has a representation that allows assertions to be made available to agents and clients.

2.9.2.2 Aggregated Resources

An Aggregated Resource is a resource that is part of an Aggregation as a result of an assertion in a Resource Map that describes the containing Aggregation.

2.9.2.3 Resource Map (ReM)

A Resource Map is a type *ore:Resource* resource with information content in which the assertions MUST describe a single Aggregation, MUST enumerate the constituent Aggregated Resources and MAY include additional properties about the Aggregator and Aggregated Resources. There are two types of Resource Maps: the Authoritative Resource Map and the Non-Authoritative Resource Map. An Authoritative Resource Map is one that is accessible via a dereference of the URI of the Aggregation it describes. A Non-Authoritative Resource Map is one that contains assertions about a URI without any reference from information obtained via a dereference of URI.

2.9.2.4 Proxy

A Proxy is an abstract entity that makes it possible to show a resource within the context of a specific Aggregation [1]. Proxies are of type *ore:Proxy*. The type is associated with the resource via an assertion in a Resource Map. The use of a Proxy is implicit. For a detailed explanation about the OAI-ORE data model, see Lagoze et al. [43].
The OAI-ORE protocol is used in many applications [27], for example, JSTOR, a USA based online academic journals archive, uses OAI-ORE in results visualisation and topology navigation tools; and The DANS (Data Archiving and Networked Services) based in the Netherlands uses OAI-ORE to improve it's "Durable Enhanced Publications" by providing durable access to value-added services and datasets [32].

2.10 Simple Web-service Offering Repository Deposit (SWORD)

SWORD is a lightweight protocol for depositing content from one location to another [35]. This protocol is a profile of the APP, and its aim is to lower barriers to deposit. SWORD's main focus is on depositing content into repositories, but this can potentially be used to deposit content into any system that is willing to receive it [35].

2.10.1 Improvements

SWORD builds on to APP by providing support for: package support, mediated deposit, developer features, auto-discovery, error documents and nested service description [2].

2.10.1.1 Package Support

The use of MIME types by the AtomPub to describe the data encoding of resources does not efficiently deal with compound types (e.g. METS packages, .tar/.zip archive files). In order to improve this SWORD introduced the <sword:acceptPackaging> element, which states the packaging formats accepted by a repository. This element is repeatable.

2.10.1.2 Mediated Deposit

Mediated Deposit allows an authenticated user to deposit a resource of which he/she/it (person or software) is not the owner. This is represented (by the client) by sending an HTTP header field *X-On-Behalf-Of* which indicates who the owner of the resource is and who made the deposit on his/her behalf. Mediated Deposit may present some security concerns as the user may not be who/what it claims to be [2] and therefore other/external security measures should be implemented.

2.10.1.3 Developer Features

SWORD aims to lower the cost involved in implementing and configuring clients and servers, and to achieve this it includes extensions that are recommended to facilitate development. The extensions are:

1. No-Op(Dry Run) - allows clients to test a server's implementation without creating a resource;
2. Verbose Output - allows servers to send to the client a detailed logging output on actions performed; and
3. Client and Server Identity - allows servers to record both server and client software identities in Atom Entry Documents.

2.10.1.4 Auto-discovery

SWORD recommends that server implementations use a $<link>$ in the $<head>$ element of HTML documents, to assist with service discovery. AtomPub makes no recommendations in this regard.

2.10.1.5 Error Documents

The SWORD profile added a new class of documents to AtomPub, that allows the server to describe error messages in a way that is more detailed than what AtomPub currently provides with HTTP.

2.10.1.6 Nested Service Description

Nested Service Descriptions allow a server to nest SWORD service definitions using the <sword:service> element. Some servers will even show the hierarchy structure within the collections. This deals with the problems faced by AtomPub when dealing with a server system that has an extremely large number of collections.

2.10.2 How does SWORD work

SWORD is a profile of APP. It adds some features while restricting others in order to provide the best possible functionality. It is dedicated to deposits and does not support all of APP's functions [35]. With SWORD, depositing is a two-stage process. It facilitates a user's action in terms of depositing, including archive files [2]. First an authenticated user sends a request to the implementation of the service document. This returns a list of collections in which the user is allowed to make deposits. After receiving the service document the user can then deposit files into the collection. The lack of authentication and unacceptable file format can cause the repository to send an error report, but if all goes well the repository sends a successful message. This can only be done in repositories that support SWORD.
SWORD can be used to facilitate e-Learning applications. As represented through examples by Sarah Currier [16], SWORD can be used in a drag-and-drop desktop tool, bulk deposit for sharing metadata, deposit from a content creation tool or drag-and-drop news feed resources into a repository. For more popular applications SWORD has experimental tools - OfficeSWORD allows direct deposit from within any Microsoft Office document and A facebook application allows a user to make a deposit while logged into facebook [16].

2.11 Interoperability Research

Some of the high-level interoperability research projects carried out in the last couple of years to provide solutions to a range of interoperability issues, as well as experimenting with and extending existing protocols, are discussed below.

2.11.1 The Kahn-Wilensky Framework (KWF)

Kahn-Wilensky is an open architecture that supports a large and extensible class of distributed digital information services, for example digital library services [36].
This architecture provides a naming principle, and a service that uses those names for the identification and location of digital objects, as well as providing an object access protocol.

All repositories must provide support for the Repository Access Protocol (RAP). This is the protocol used by originators for depositing and accessing objects in repositories. The system functions as follows [36] [37]:
1. A user with digital data creates a digital object. A digital object is composed of digital data and a unique identifier called a *handle*. This is done by sending a request to an authorised handle generator.
2. The user deposits the digital object into one or more repositories.
3. The digital object handle and the repository name or IP address is registered to a global system of handle servers.
4. Finally the users can send a handle of a registered digital object to a handle server in order to retrieve the name or IP addresses of the repositories storing a specific digital object.
The three services defined to perform the above mentioned actions are: ACCESS_DO, DEPOSIT_DO and ACCESS_REF. ACCESS_DO and DEPOSIT_DO are used for accessing and depositing objects and ACCESS_REF is used to access repository reference services [23].
Kahn-Wilensky's underlying architecture forms a base for extensions that can be customised for information of various formats [6]. In this framework the representation of metadata digital objects influenced the Making of America II project which later gave rise to Metadata Encoding and Transmission Standards (METS) [56]. Other contributions of the KWF include promoting handles as global unique identifiers and defining the relationship between digital objects and repositories.
Some examples of implementations based on the Kahn-Wilensky architecture are: the Interoperable Secure Object Stores (ISOS), and the Dienst protocol [23].

2.11.2 The Dienst Protocol

Dienst is an architecture and protocol for digital libraries across multiple servers [19].
Initially called the Computer Science Technical Report Project, this ARPA funded project originated from the need to create a digital library of Computer Science technical reports [18].
The protocol supports the following individually defined and distributed services[17]: an *info service* that provides information about the state of a server; an *index service* that processes queries and returns a list of matching record identifiers; a *repository service* that stores digital documents, each with a unique id also called DocID; a *query mediator service* that sends queries to the appropriate index servers; an *index service* that processes queries and returns a list of matching records identifiers; a *collection service* that provides information on the services interaction to form a logical collection; and a *registry service* that stores information about users.
Additionally there is a *user interface service* for interaction between the above mentioned services and their protocols [17].
The services are defined individually, and when combined they create a distributed digital library that provides functionality for deposit and storage of digital resources, as well as access to those resources by discovery and browsing [17].
Dienst requests are called verbs. Each service supports a set of verbs and one service can support different versions of one verb. Requests are sent in the URL via the HTTP

protocol. If a service receives a verb with a version it does not support, it must return an error. The responses are HTTP responses, which can be of MIME type *text/plain*, *text/xml* or *text/html* [17].
The Dienst protocol not only provides a conceptual architecture for communications with services in distributed digital libraries but also the software system that implements the protocol [14].
Over one hundred institutions used Dienst to be part of the Networked Computer Science Technical Reference Library (NCSTRL) but, with the creation of OAI-PMH, many of them have transitioned to the latter. The design of OAI-PMH was largely based on improvements made by considering the lessons learned from the design of Dienst, for example Dienst supports over 30 verbs and OAI-PMH only supports 6 verbs [31], and a harvesting approach is preferred to a cross archive searching approach because it avoids the problem of not getting updated results (possible in federated search if one or more repositories are down), by collecting and storing all data in a central location.
In 2003 a Dienst OAI-PMH Gateway (DOG) was created to allow OAI-PMH harvesting from existing and at risk Dienst repositories [31].

2.11.3 Simple Digital Library Interoperability Protocol (SDLIP)

SDLIP is a protocol that defines simple interfaces for interoperability between data providers [29]. It was part of a Stanford University project called Stanford Digital Libraries Technologies [85]. SDLIP's main goals were the simplification of both client and server side implementations; server support for stateful and stateless operations; dynamic load balancing for server; support for thin clients; and implementations via both distributed object technology (CORBA and HTTP/CGI) [85].
SDLIP operations are divided across three interfaces, namely: the *Search* interface that allows the submission of search queries, the *Result Access* interface that allows the client applications to access results from a search request and the *Source Metadata* interface that allows clients to question a library service proxy about its capabilities.
At the very minimum, a SDLIP server should implement the Search interface.
There are two different levels of capabilities:
SDLIP-Core: supports only the implementation of synchronous operations. Clients send search requests and wait until the server responds.
SDLIP-Asynch: allow clients to send search operations and have responses returned immediately.

SDLIP-Core can be configured to work in different ways, for example [85]:

1. Requests are submitted synchronously via the search interface and the results are returned as part of a call.

2. The server stores the result set of a clients' first request, at least for a specific period, and it is used again if the client sends another synchronous request for the same result set.

3. An overloaded service object may delegate interactions with the client to another service object.

SDLIP-Asynch is an extension of SDLIP-Core that allows servers to interact with clients asynchronously, and includes the implementation of a client side delivery interface. This interface is used to return search results immediately, one by one as the became available or in batches [84].
Using SDLIP, clients can customise a request to return a specified number of documents and also which parts of the document to return, for example request only the title and authors for 10 documents [85].

2.11.4 The Warwick Framework

The Warwick Framework is a container architecture for aggregating distinct metadata packages. Is the result of the Metadata II Workshop that took place in the April 1996, in Warwick (United Kingdom). This workshop was a follow-up to the 1995 Metadata Workshop in Dublin, Ohio (United States of America), that resulted in the creation of the Dublin Core metadata standard.
The scope of the Dublin Core metadata format was intentionally limited to avoid "the size and complexity of the resource description problem" [40] [41]; But it soon became clear that by avoiding that problem there was room for other problems. The Warwick workshop was centered around 3 main questions [40]:
1. Should the number of elements in the Dublin Core be expanded or contracted?
2. Should the syntax of the Core be strictly defined or left unstructured?
3. Should the Core be targeted solely at the existing WWW architecture, or extend that architecture?
Analysis and discussion of many factors led to the creation of the Warwick Framework. The framework has two main components: containers and packages [41].
A container is the unit for aggregating the typed metadata packages/sets. A package is a typed object [20]. A container may hold packages managed and maintained by distinct parties.

2.11.5 The OpenDLib Model

OpenDLib is an expandable software package that allows for the creation and management of digital library systems customised to specific community requirements. The model specifies the configurations of a OpenDLib digital library system [61] [12]. Configurations that comply with the model are known as OpenDLib legal configurations.
The OpenDLib architecture model uses three concepts [12] [61]:
1. Services - interact to provide functions for the coordination of different tasks. They can be centralised , distributed and/or replicated across multiple federated servers. Communication among services is regulated via the OpenDLib Protocol (ODP)[10].
2. Server - manages resources shared over the network.
3. Region - an abstract concept that consists of centralised and distributed service instances as well as a set of instances of replicated services (one of each service type).
OpenDLib can be expanded to include new services to support new functions that will allow for even more customisation of the digital library according to the community

[10]The ODP protocol is an extension of the Dienst Protocol

needs. All extensions are done in a "plug-and-play" way, i.e. without having to deactivate the library [12]. Expansions are supported through three key mechanisms, namely: configurable services, open architectural infrastructural and basic utility services [12].
Flexible systems like the OpenDlib model are a perfect platform to accommodate a suite of protocols that provides various services, such as the one proposed in this research.

2.11.6 Driver and Driver II Infrastructures

The Digital Repositories Infrastructure Vision for European Research projects DRIVER and DRIVER II - were funded by the European Union for the creation of a cohesive, robust and flexible digital repository infrastructure, to offer sophisticated services that would connect the Global users to research output from European institutions [90].
The infrastructure works by creating a virtual integration of multiple repositories all over Europe.
DRIVER was the first phase of the project, that resulted in the development of a test-bed search portal called D-Net. The portal provided Open Access content from over 70 institutions[90].
DRIVER II is the second phase, which involves the production of a full quality system that will include content from even more institutions than those available in phase 1 and promote greater visibility in the global Open Access repository scene [90].
These affords have resulted in the creation of the Confederation of Open Access Repositories (COAR), in 2009, and DRIVER reports are now available in the series of international publications on Trends in Research Information Management (TRIM) [90].
Much like in this research, DRIVERs main goal is to enhance interoperability between data and service providers, while providing the required functionalities to the end users [22].
This infrastructure uses the OAI-PMH protocol for harvesting, and an extension of OAI-PMH called OAI-SQ for searching.

2.11.7 Open Digital Libraries (ODLs)

Open Digital Libraries is an extension of the OAI-PMH protocol, that involves a framework of service components for building extensible digital libraries [88].
The ODLs approach proposed a framework of individual component DLs, that could each be customised to provide one or more specific interoperability services, and as a whole work as a network of extended open archives [88].
The framework supports the 6 OAI-PMH verbs and its own ODL protocols for other services, namely: ODL-Union, ODL-Filter, ODL-Search, ODL-Browse, ODL-Recent, ODL-Annotate, ODL-Review and ODL-Submit [89].
Case studies of the individual service components showed the feasibility of the construction of DLs in a simple and and repeatable manner [87].
This is a multiple service support framework, a model similar to the one proposed in this research.

2.11.8 OpenSearch

OpenSearch is a collection of simple formats for sharing search results [62]. It provides a way for Websites and search engines to publish their search results in a standardised format. Formally it only supports RSS and Atom but it also accepts other formats, e.g. HTML. The framework provides support for the features described below[97]:

1. OpenSearch Description files - are XML files used to identify and describe the Web interface of search engines. OpenSearch is supported by a number of search engines, for example Bing, Arora, Alfresco, Gnome (Do and Shell), Google Chrome to name a few.

2. OpenSearch Query Syntax - this syntax state where to retrieve search results from.

3. OpenSearch RSS - also know as OpenSearch Response it is a format for providing OpenSearch results.

4. OpenSearch Aggregators - are sites that can display OpenSearch results.

5. OpenSearch Auto-discovery - lets the user know that there is a plugin link.

This combination of multiple features support allow clients to learn about the user interface of search engines and provides support for search engines to add search metadata to results in a variety of formats.

This chapter presented a brief introduction to interoperability protocols, an analyses on the features and functionality of some of the existing protocols and discussions about various research interoperability projects.

Chapter 3

User Survey

The best way to obtain information on what users want and what is best for them is by getting that information from potential users. That is the approach taken for this research.
A user survey was conducted, to get information from protocol implementers/users about the state of currently used protocols and how, if at all, they could be improved.
The method chosen to conduct this survey was an online questionnaire. The participants were asked to answer 6 questions about various protocols (see appendix C).

3.1 Sample Population

To get a diversified target population of possible survey participants, people with interests in different areas of interoperability and different protocols were invited to take part in this survey. Invitations to participate were sent to the following mailing lists: SRW, eprints-tech, dspace-tech, owner-atom-protocol list, ore-implementers, oai-implementers, rss-public and the Computer Science postgraduate students at the University of Cape Town (UCT)[1].
A total of 30 participants took part in the survey. Seven survey responses were not used because they were incomplete and only contained an answer to question 1.
Below is a summary of the findings obtained from the survey responses of 23 participants:

3.2 Most Popular (Most Used and/or Known) Protocol

Participants were asked to rate their level of expertise with each of the protocols according to the following criteria:

- Level 1:Expert implementer
- Level 2:Implemented (have written code for an implementation of the protocol)

[1]the addresses of the mailing lists are as the following: srw@mail.dei.unipd.it, eprints-tech@ecs.soton.ac.uk, dspace-tech@lists.sourceforge.net, owner-atom-protocol@vpnc.org, ore-implementers@openarchives.org, oai-implementers@openarchives.org, rss-public@yahoogroups.com and grads@cs.uct.ac.za

- Level 3:Read and understood
- Level 4:Heard about it, but do not know the details
- Level 5:Never heard about it

Table 3.1 shows the number and percentage of people according to their knowledge and level of expertise of different protocols.

Protocols	**Number of People at Each Level of Expertise**					
	1	**2**	**3**	**4**	**5**	**No answer**
RSS	3 (13%)	6 (26.1%)	9 (39.1%)	3 (13%)	1 (4.3%)	1 (4.3%)
ATOM	1 (4.3%)	4 (17.4%)	7 (30.4%)	9 (39.1%)	1 (4.3%)	1 (4.3%)
APP	0 (0%)	1 (4.3%)	1 (4.3%)	3 (13%)	17 (73.9%)	1 (4.3%)
Z39.50	0 (0%)	0 (0%)	6 (26.1%)	12 (52.2%)	4 (17.4%)	1 (4.3%)
OAI-PMH	7 (30.4%)	7 (30.4%)	5 (21.7%)	0 (0%)	3 (13%)	1 (4.3%)
OAI-ORE	1 (4.3%)	1 (4.3%)	5 (21.7%)	9 (39.1%)	6 (26.1%)	1 (4.3%)
SRU/W	0 (0%)	0 (0%)	7 (30.4%)	5 (21.7%)	10 (43.5%)	1 (4.3%)
SWORD	0 (0%)	0 (0%)	7 (30.4%)	5 (21.7%)	10 (43.5%)	1 (4.3%)

Table 3.1: Participants' Levels of Expertise of the Protocols

Findings:

- **OAI-PMH** - is the most popular protocol, with 60.9% of participants (14 people) as experts (levels 1 & 2).
- **RSS** - is the second most popular protocol, with 39.1% of participants (9 people) in levels 1 & 2.
- **ATOM** - is in third place with 21.7% of participants (5 people) in levels 1 & 2.
- **OAI-ORE** - only 8.7% of the participants (2 people) are OAI-ORE experts.
- **APP** - none of the participants are APP experts and only 4.3% of the participants (1 person) are in level 2.
- **Z39.50** - there were no experts for the Z39.50 protocol, and only 26.1% of the participants (6 people) have read and understood this protocol.
- **SRU/W** - there were no experts of the SRU/W protocol, and only 30.4% of the participants (7 people) have read and understood the protocol.
- **SWORD** - just like SRU/W, there were no experts of the SWORD protocol, and similarly there were only 7 people who have read and understood the protocol; 43.5% of participants never even heard of the protocol.

3.3 Most Unpopular (Least Known/Used) Protocol

The data in the above section suggests that:

- **APP** is the least known protocol, with 73.9% (17 people) who have never heard of it.
- An equal number of people, 10 people or 43.5% per protocol, have never heard of **SRU/W** and **SWORD**.
- 13% of participants, (3 people), never heard of **OAI-PMH** and,
- Only 4.3% of participants (1 person per protocol) have never heard of **RSS** and **ATOM**.

3.4 Most Useful Features

The participants named the features below as being the most useful features for the different protocols.

3.4.1 RSS

- Aggregation
- Flexible namespace use (dc, PRISM)
- Flexible use areas (not restricted to only one area of use, e.g. can be used by non-librarians)
- Link association with date
- Popularity (Its popularity attracts more people, trust)
- Sharing of quick text information and links
- Simplicity (easy to learn and implement)
- The use of popular standards (XML)
- Time saving (only get information on your interests)
- Value added services (podcasts)

3.4.2 ATOM

- Aggregation
- Clear data model
- Flexible use areas (not restricted to only one area of use, e.g. can be used by non-librarians)
- Link association with date
- More standardised (e.g. fewer versions) and has better specifications than RSS
- Multiple formatting (dc, PRISM)
- Powerful
- Simplicity
- The use of popular standards (XML)

3.4.3 Z39.50

- Allows federated search
- Provides library data interchange
- Standard for a particular community (e.g. libraries)

3.4.4 OAI-PMH

- Allows community based aggregation of metadata
- Flexible metadata formats
- Integration into software packages (Dspace, CONTENTdm)
- Harvesting of vast amounts of data at once (saves time)
- Low server load
- Platform independent metadata retrieval
- Resumption tokens allow users to download on their own schedule
- Simplicity in implementation
- Specifically designed for digital repositories
- Strings open archives together
- Very good at generically providing db data

3.4.5 OAI-ORE

- A solution to the aggregation problem
- Platform independent metadata retrieval

3.4.6 SWORD

- Interoperability with other systems (e.g. CRIS); possible upload to repository through MS word
- Multiple deposit methods
- Platform independent repository deposit
- Reduces work load (multiple deposits at once)

Many features were mentioned by the users as most useful for the different protocols, but amongst that a few stand out for being mentioned as useful features in more than one protocol. These features are: simplicity, aggregation and platform independence. The use of popular standards, flexibility, multiple metadata formats and time saving were also amongst the top favourite features mentioned.

3.5 Least Useful Features

The participants have also indicated which features they find least useful and/or the shortcomings in each protocol.

3.5.1 RSS

- Lacks features (most features are extended modules)
- Lacks semantics
- Poor documentation (often leads to bad implementations)
- Poor specifications of content type (programmers use it for different things)
- Too many versions

3.5.2 ATOM

- Lacks semantics
- Too heavily geared towards blog posts, therefore it is misappropriated in systems interoperability

3.5.3 APP

- Overly zealous in its requirements

3.5.4 Z39.50

- Complex/arcane
- Heavy server load
- Outdated
- Use restricted to a specific community only (librarians)

3.5.5 OAI-PMH

- Assumption of item-level description
- Cumbersome syntax and operation
- Lacks variables (e.g. searching function)
- ListIdentifiers (no further details were provided on this issue)
- Multiple metadata formats make the protocol harder than it needs to be
- Sets are chaotic (no further details were provided on this issue)
- Underutilised (can be used for much more than transporting metadata records, it can transport an entire archive including the digital signatures)

3.5.6 OAI-ORE

- Complex
- Sets are chaotic (no further details were provided on this issue. Note that the user made a mistake with this comment as sets don't exist in OAI-ORE)

3.5.7 SRU/W

- Complex to implement
- It is difficult to translate CQL to backend search engines

3.5.8 SWORD

- Does not allow withdrawals (restricted to deposits only)
- Relies exclusively on packaging for content passing
- Unfinished (still evolving)

One participant said that RSS does not have any feature which is not useful, while 2 participants said the same about OAI-PMH.
The top features mentioned as least useful are: lack of features, semantics issues and complexity and quality of documentation/specifications. Also interesting is the fact that while multiple metadata formats was mentioned as one of the most useful features for more than one protocol, it was mentioned as one of OAI-PMH's least useful feature by some participants - in fact more participants than the combined number of those who said that it is one of the most useful features. That emphasizes the idea that these choices are subjective.

3.6 Suggested Improvements

The list below indicates the participants' opinions on what improvements can be made to the different protocols in an attempt to improve on them.

3.6.1 RSS

- A standard interpreter with graphical user interface
- The ability to query specific time frames (a layer above the protocol)
- Adding new tag (element) by the user as needed

3.6.2 ATOM

- The ability to query specific time frames (a layer above the protocol)
- Adopting some of the semantics of OAI-PMH

3.6.3 Z39.50

- Allow full harvest
- Combined with SRU, Z39.50 becomes simpler

3.6.4 OAI-PMH

- Provide services other than metadata extraction (e.g. negotiate the legal issues and allow access to the actual data)
- Consider persistent resumptionTokens
- Definition of sets restriction (nomenclature, hierarchy)
- Although interoperability is the key issue, other aspects of the protocol are also important (e.g. efficiency)

3.6.5 OAI-ORE

- Consider persistent resumptionTokens (Note that the user made a mistake with this comment as there are no resumptionTokens in OAI-ORE)
- Definition of sets restriction e.g. nomenclature and hierarchy. (Note that the user made a mistake with this comment as sets don't exist in OAI-ORE)

3.6.6 SRU/W

- Allow full harvest and simplify

3.6.7 SWORD

- Better integration with OAI-ORE
- Extension for non-packaged material
- Implementation of replace and add functions

3.6.8 All Protocols

- Although interoperability is the key issue, other aspects of the protocol are also important (e.g. efficiency)
- Better documentation
- Create a protocol for the WWW rather than one for a specific community
- Do not require data that is not essential
- More standardised namespaces, to better tag data
- Simplicity equals a protocol that is easier to be widely adopted
- There should be a protocol with a standardised interface that allows harvesting, syndication and searching with a very small set of mandatory operations/parameters (for the sake of simplicity)

A number of improvement suggestions were given for the different protocols, same of which stand out for being mentioned for more than one protocol, for example: the ability to query a specific time interval and to add new tag elements in both RSS and ATOM; persistent resumptionToken and definition of set restrictions for OAI-PMH; and simplicity for Z39.50 and SRU/W.
Suggestions for all protocols were along the lines of simplicity, efficiency, better documentation and standardisation.

3.7 General Comments

The participants were also given a chance to make any comments related to the issue of protocols and interoperability that they felt were appropriate for the purpose of the survey, and to help protocol designers better understand their possible target users needs. The comments made are:

- RSS and Atom do not perform well for interoperability on bibliographic applications.
- Standards are the only way to make interoperability work. Using these standards, it is possible to upgrade systems and change systems completely (Dspace to Fedora) without losing expensive work.
- There is much more work going into developing protocols for interoperability than there is work on the interoperability of the content. All of these protocols deal with the transport of some kind of data, but most do not concern themselves with the data itself. But data interoperability is the core of the interoperability problem.

3.8 Survey Conclusions

Based on the population sample it is believed that OAI-PMH is classified as the best-known protocol in this survey based on the fact that most of the respondents are OAI-PMH implementers, which means that OAI-PMH is mostly known by its implementers only. RSS and ATOM, on the other hand, are well known to both implementers (levels 1 2) and the general Web users (levels 3 4). There was only one person (level 5) who never heard of either RSS or ATOM, compared to 3 people who never heard of OAI-PMH.
There was some disagreement on the issue of whether multiple metadata formats for OAI-PMH was a good or bad feature; overall most participants said that multiple metadata formats make things harder.
Although the number of participants in this survey represents a very small sample of the "interoperability world population", the data obtained was very useful for the purpose of this research, and from this it is clear that there is still room for improvement in the existing interoperability protocols.
The results of this survey led us to conclude that a good interoperability protocol should be: simple enough to allow programmers to implement, explore and experiment while requiring only operations that are crucial to the performance of the protocol, but robust enough to provide value added services for the data extracted. The perfect solution if perfection can ever be achieved, is to design protocol services that combine the level of simplicity and lightness of RSS, with the quality of the structure and semantics of OAI-PMH and the level of efficiency of SWORD.
In order to try to find potential solution(s) a set of experimental protocols was designed to be evaluated against the existing ones. The results are expected to suggest possible improvements for future protocols.

Chapter 4

Design

An analysis of the survey results provided insight into the user needs.
Based on these results two main focus points were chosen as the key design goals: simplicity and efficiency. This was an experimental design, that could possibly show an alternative approach to protocol design.

4.1 Introduction

Given the survey results on the current set of protocols and the areas of possible improvement, an experimental protocol called High-level Interoperability Protocol - Common Framework "HIP-CF" is proposed.
HIP-CF Protocol is a high-level application layer interoperability protocol for locating and retrieving digital data and metadata records. This protocol aims to facilitate access to digital resources by making them visible to Web agents and providing support for browse, harvesting and search of resources.
The base protocol and the services it supports are presented below, in the following order: HIP-CF Preamble, Xsearch, Xharvesting and Xbrowse.

4.1.1 Design Approach

Following the simplicity and efficiency goals, the approach chosen for the protocols design was a "ground-up minimalistic design approach". What that means is, starting with basic and absolutely necessary support (i.e. the use of HTPP and XML), and then only add enough features that are needed for an efficient protocol, as opposed to adding every possible combination of features.
The most obvious drawback from this approach is the exclusion of features that may be considered important by potential users [63]. Which features to exclude is definitely a difficult decision for the designer, especially because excluding the wrong features may lead to the design of a protocol that does not met the needs of the users [63]. Nevertheless it is possible considering that only a fraction of the features supported by systems with big and complex specifications are used by the majority of users [63].

4.2 High-level Interoperability Protocol - Common Framework (HIP-CF)

The High-level Interoperability Protocols Common Framework (HIP-CF) is a common framework that facilitates high-level interoperability between heterogeneous systems. HIP-CF has different interoperability protocol services built as a layer on top of the common framework, thus creating a suite of protocols. The protocols use HTTP for data transfer and follows the HTTP specifications as stated in RFC 2616[24]. The services supported by the HIP-CF suite are Xbrowse, Xharvester and Xsearch.

4.2.1 Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in the HIP-CF documents are to be interpreted as described in RFC 2119[10].

4.2.2 Terminology

The terms presented in this section are used to define the roles and/or actions involved in the functionality of the protocol[1]. The terminology is presented in alphabetical order and not in order of occurrence or use.
Client - A computer connected to a network that makes a request for resources or services from a server located elsewhere on the network[68].
Data Source/Repository - A system that exposes its contents and allows access to it by external sources.
Digital Library - A managed collection of digital information, with associated services, where the information is accessible over a network [5].
Metadata - Data that describes data, for example metadata about a journal article can include creators_name, publishing_date, abstract, etc.
Query - A request made to a computer system (e.g. database, digital repository) to retrieve a particular set of data records[68]. A query can have one or more parameters.
Record - A data structure consisting of a collection of fields, each possibly containing a different data type[68]. A record here refers to a metadata record presented in XML.
Request - A message that requires some form of reply from its recipient[68].
Response - A document returned by the server as a result of a client request.
Server - A computer or computer program that is designed to provide shared services to other computer systems on a network.
URL - The Uniform Resource Locator is a string of characters used to identify a resource on the Internet (e.g. a Web page, a server, a file)[5].
XML - Extensible Markup Language is a set of rules for encoding structured documents in a machine-readable form[2]. XML structures and stores data[3]

[1]Unless otherwise indicated by a reference, the terms used in this document are defined according to their specific use in or for HIP-CF.

[2]http://en.wikipedia.org/wiki/XML

[3]An alternative to using XML is to use JSON, a lightweight data-interchange format based on JavaScript (see www.Json.org).

4.2.3 HIP-CF Principles and Guidelines

Implementations of HIP-CF services MUST comply with the principles and guidelines below:

4.2.3.1 Simplicity

HIP-CF implementations **SHALL** be as simple as possible. The protocol is designed to be efficient and reliable but, most importantly, to be simple, as opposed to being complex and computationally expensive. The motto is "do what needs to be done in the simplest possible way, and add complexity only when or where strictly necessary". A bottom-up approach is **RECOMMENDED** for the implementation of all protocol services. In other words: start from nothing, and build up the implementation until you achieve the desired functionality and efficiency. Use the simplest available solutions and avoid any non-crucial elements.
Simplicity is important because it is a key factor to reduce costs, save time and possibly increase compliance with the protocol specification.

4.2.3.2 Robustness

HIP-CF implementations **MUST** be robust, i.e. have the ability to recover from invalid input data and other error conditions, as well as operate in adverse conditions.

4.2.3.3 Data Storage

There are no requirements for where or how data should be stored. It is left to the organisation or individual implementing the protocol to choose a solution that best suits their needs. Data sources **MUST** ensure that the repository/system is configured to allow clients to find and access the resources available in the repository.

4.2.3.4 Reuse

HIP-CF implementations **MUST** take advantage of existing technologies and standards such as HTTP , XML, JSON and REST.

4.2.4 HIP-CF General Model

The framework provides a consistent and extensible standardised structure (vocabulary, principles and guidelines) (see Figure 4.1) that is based on the Representational State Transfer (REST) architecture, and allows the implementation of different high-level interoperability services.

4.2.4.1 Protocol Model

The protocol has two main characters/components: a client and a server. The client and server communicate through request-response pairs. For simplicity, it is **RECOMMENDED** that when possible HIP-CF implementations use one request-response pair

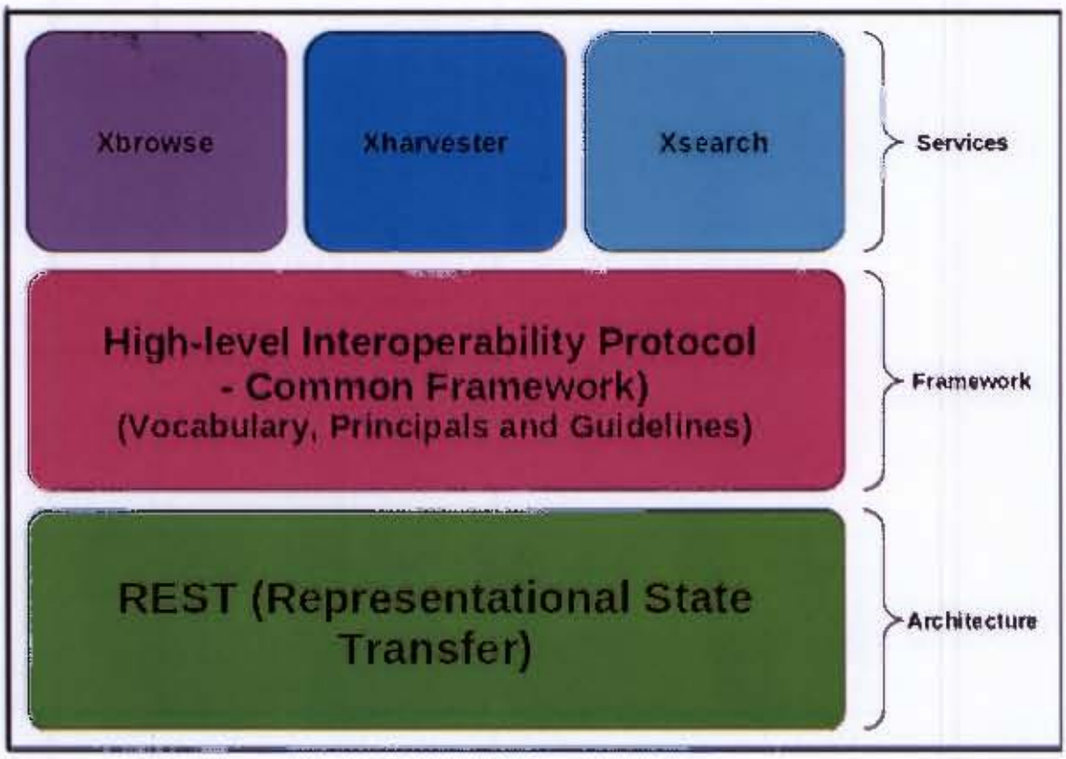

Figure 4.1: High-level Interoperability Protocol - Common Framework

for records retrieval as opposed to sending a request to find out if there are matching records and then sending a second request to retrieve the actual records.

4.2.4.1.1 Client The client is the application that is developed at the client side to request data access from the server or data source. The client initiates the connection between client and server by sending a request for services from the server (see Figure 4.2).

4.2.4.1.2 The Server The server is the application on the data source side. The server receives a request, processes it and sends a response back to the client (see Figure 4.2). A response page may contain matching record(s), no match found response, resource no-longer available, an HTTP error, or other appropriate responses.

4.2.4.1.3 The Application Layer Protocol The RECOMMENDED protocol to transfer the request and response pairs between the client and the server is the HyperText Transfer Protocol (HTTP) (see Figure 4.2). HTTP is an application layer protocol for distributed, collaborative and hypermedia information systems [24].
Requests can be sent using either one or both HTTP GET and HTTP POST. Special characters in the URL (e.g. space, $, <, :) should be encoded according to URL encoding standards [9].

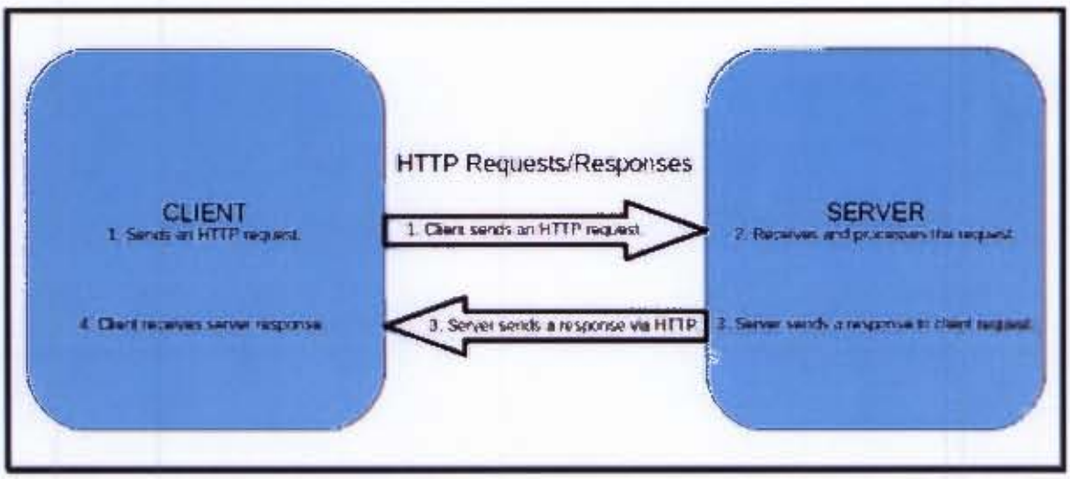

Figure 4.2: HTTP Request and Response Model

Using HTTP GET the parameters are sent via the URL. The parameter=value pairs are separated from the URL by a question mark and from each other by an ampersand. For example: http://www.serverexample.com?parameter1=one&parameter2=2.
Using HTTP POST the arguments are carried in the message body, and there is the advantage of not limiting the length of the arguments. For example:
Post http://www.serverexample.com
Content-Length: 128
Content-Type: application/x-www-form-urlencoded
Request=parameter1=one&parameter2=two
It is **RECOMMENDED** that whenever possible customised messages and solutions for HTTP errors are used, that is to help the client better understand errors and find solutions. HTTP error messages may also be used to control data traffic, for example error code 307 can be used for temporary redirect of a request to a different URI.

4.2.4.2 Parameter Model

Each protocol service defines a set of parameters. There are the mandatory parameters, (parameters that MUST be implemented), and optional parameters (parameters that improve the quality of the results but are only used by choice).

4.2.4.3 Error Messages

When a request generates an error the server responds by sending the client an error message. While an error message **MAY** contain both machine and human readable formats, at a minimum it is **RECOMMENDED** that it contains an HTTP error code. Examples:
<error>
<HTTP_code>404</HTTP_code>
<description>No match found.</description>

</error>

<error>
<HTTP_code>400</HTTP_code>
<description>Bad request. Please check that there are no spelling errors in the request, and that all the parameter values are correct. </description>
</error>

4.3 Xsearch

4.3.1 Introduction

The Xsearch protocol is a high-level interoperability protocol that provides a simplistic service for clients to search for resources from digital libraries, databases and other sources of digital data.
Search is the process by which a client can retrieve an available resource from a server. To search for a resource a client sends a request or query[4] (a search term plus other optional parameter(s)) to the server. The server responds by sending back a list of the resource(s) that match the client's request, or a message to inform the client that none of the resources available on the server side match the client's request.

4.3.2 Data Transfer

Xsearch makes use of HTTP methods GET and POST to send requests to servers. Either way the server receives a request that may have a format similar to: http://address.example.com?q=paihama&start=0 "http://address.example.com" is the server's network address, "?" separates the URL from the parameters and "q=paihama&start=0" are the query parameters. The server processes the request and sends a response in a predefined format, for example XML.

4.3.3 Parameter Model

Xsearch has four defined parameters (see Table 1 below). A request has to contain the mandatory parameter queryword, and any combination of the optional parameters.

4.3.3.1 queryword

The *queryword* is the mandatory request parameter and it is made up of one or more words sent by the client in a request and used by the server to retrieve resources whose content matches the parameter.
In order to perform a search the client should send a request that contains at least the *queryword* parameter. A *queryword* can be an authors names, a keyword, a word in the abstract or any word in the text, etc. The server has control over which parts of the resources/records are searched to find a match.

[4]The words request and query will be used interchangeably.

Parameters	Occurrence	Description
queryword	mandatory	The word/phrase submitted by the client which the server uses to check its resources in order to find matching records.
rpp	optional	Records Per Page (rpp) indicates the number of records to display per results page.
start	optional	The number of the first record on the results page. For example if the records counter is initialised at 0, start=1 will return a page of results starting from the second matching record.
metadataFormat	optional	Tells the server to only return results that are in the metadata format specified by the client.

Table 4.1: Xsearch parameters and their occurrence and descriptions

4.3.3.2 rpp

The *rpp* parameter allows the client to decide the number of records the server should display in a single response page. This is only applicable when there is more than one record that matches the query. On the response page, the server tells the client how many records in total match the query.
If the number of records that match the query is higher than the number of records displayed on the results page, then the client can use the *rpp* and *start* parameters to retrieve the remaining matching records. When the results are displayed on a Web page, a paging system can be used to display all available records.
When the *rpp* parameter is not used, the server returns its default number of matching records per page starting from the first matching record. The default number is determined by the server settings.

4.3.3.3 start

The *start* parameter allows the client to make a request to the server to retrieve the records that match a query starting from a specific point within the result set. For example, if *start*=2 and the counter is set to start at 0, the server will return results from the third matching record. In the same way *start*=0 will get results starting from the first matching record, *start*=1 will get results from the second matching record, and so on. This parameter can be used with *rpp* to tell the server how many records to retrieve and which record to start at.

4.3.3.4 metadataFormat

There are a number of metadata formats that can be used to present a query's response, for example Dublin Core, MARC 21 and MODS. The choice of metadata format is related to the needs of the community implementing the service and the data source.
Xsearch does not impose the use of any specific metadata format. This is to allow different communities to create their own metadata format(s) if they feel that none of the existing formats are appropriate for them.

Communities that have no need or desire to create their own metadata standards are **RECOMMENDED** to use Dublin Core[5].
As **RECOMMENDED** by the W3C[6], if necessary, an XML namespace[7] can be created to clear any ambiguities that may exist for elements that happen to have the same name and to group common elements together.

4.3.4 Processing model

Processing occurs on the server side. The server:

- Receives a request (in a language/format that it understands);
- Decodes to get the parameter values;
- Uses the parameter values to perform a search query on it's own data source;
- Gets a response from the data source, encodes it in a format that the client understands; and
- Sends back the response.

The client also processes the results page(s) returned by the server and uses the data/matching record(s) as it sees fits. However this use of the retrieved data by the client is outside the scope of Xsearch.

4.3.5 Query Model

Any query language may be used in the implementation of Xsearch. The choice of query language will be influenced by, amongst other things, the data source.

4.3.6 Result Set Model

The result set is a list of records or an error response returned by the server as a response to a request. The result set is usually an ordered list whose format and contents are defined according to request parameter values or server settings. Result sets only contain metadata.

4.3.7 Protocol Parameter Use Examples

4.3.7.1 Mandatory Parameter Only - Request

A request made with only the mandatory parameter *queryword* can be in a format similar to this http://174.19.284.1/cgi-bin/Search.pl?queryword=paihama+poulo. If one or more matches are found this request will generate a response set containing metadata

[5]http://dublincore.org/
[6]World Wide Web Consortium. http://www.w3.org/
[7]http://www.w3.org/TR/xml-names/

about all the records that match the words paihama and poulo, any work that the two authors have done individually, together or with other authors as well as records containing the words paihama and/or poulo in any metadata field used for the search.

Mandatory Parameter Only - Response

<results>
<total_matches>4</total_matches>
<record>
<title>
Survey of the Effectiveness of Current Interoperability Protocols, A
</title>
<creator>Paihama Jorgina Kaumbe do Rosario</creator>
<creator>Suleman Hussein</creator>
<description>
Interoperability is the capability of different systems to communicate and exchange data with one another, using a set of predefined formats and protocols that will allow the systems to use one anothers services successfully.
</description>
<keyword>
interoperability, interoperability survey, interoperability protocols
</keyword>
<event_dates/>
</record>
<record>
<title>
Preserving Endangered Languages using a Layered Web-based Archive
</title>
<creator>Poulo Lebeko B. N.</creator>
<creator>Paihama Jorgina Kaumbe do Rosario</creator>
<creator>Mohamed Nour Marwan I. M. E.</creator>
<creator>Suleman Hussein</creator>
<description>
Many human languages, an essential part of culture, are in danger of extinction.UNESCO estimates that at least a half of the world's 6500 spoken languages will disappear within the next 100 years.
</description>
<keyword>
Language Preservation, Digital Repository, User-Centred Design, User Interfaces
</keyword>
<event_dates>September 2, 2009 September 4, 2009</event_dates>
</record>
</results>

4.3.7.2 Mandatory and Optional Parameters - Request

A request with the mandatory parameter *queryword* and a combination of optional parameters, for example *start* and *rpp*, helps fine-tune the result set to the client's needs.

The request can be in a format similar to http://174.19.284.1/cgi-bin/Search.pl?queryword =paihama&start=0&rpp=1. If a match is found, this request will generate a response set containing metadata about one record that matches the word paihama. The record returned will be the first record that matches the query.

Mandatory and Optional Parameters - Response

<results>
<total_matches>3</total_matches>
<record>
<title>
Survey of the Effectiveness of Current Interoperability Protocols, A
</title>
<creator>Paihama Jorgina Kaumbe do Rosario</creator>
<creator>Suleman Hussein</creator>
<description>
Interoperability is the capability of different systems to communicate and exchange data with one another, using a set of predefined formats and protocols that will allow the systems to use one anothers services successfully.
</description>
<keyword>
interoperability, interoperability survey, interoperability protocols
</keyword>
<event_dates/>
</record>
</results>

4.4 Xharvester

4.4.1 Introduction

Xharvester is a high-level interoperability protocol that provides a simplistic service for clients to harvest metadata records from various sources of digital data.
Data harvesting is the process by which a client accesses a server's data repository and collects metadata records.
The process of harvesting data involves two main components; a data provider also called a server and a service provider also called a harvester.

4.4.2 Data Provider

A Data Provider is a network-accessible server that stores and exposes data, and allows client applications to access their data. A Data Provider can be any application that stores digital data, for example a digital library or a database. The data access is usually restricted to metadata only.
A data repository contains many resources. A resource will have one or more metadata records associated with it, for example: a published paper named Computer Science 101

is a resource. That resource can have metadata records in different formats, such as Dublin Core, MARC21[8], LOM, METS and MODS[9].
Communities can create their own style of metadata or conform to one of the existing styles. However for the purpose of facilitating interoperability, the use of unqualified Dublin Core is **RECOMMENDED**.
Metadata records that point to the same resource will have the same unique identifier. A unique identifier is an identifier that will point to one specific resource. Each identifier is unique in the sense that it will point to one and only one resource, but all records that point to that one resource will use the same identifier. In order to distinguish between records in different metadata formats but relating to the same resource the harvester will use the metadataFormat parameter, which is discussed in section 3.
To create unique identifiers data providers may follow the "Uniform Resource Identifiers (URI): Generic Syntax" [9], or create a unique identifier scheme that suits the community needs.

4.4.3 Service Provider

A Service Provider or a harvester is an application that accesses, collects and stores metadata from one or more data providers. Data harvesting is performed at a set interval, for example, repository X is set to harvest repository Y on a daily basis at 12h00. This is done so that the harvester always has updated records of the resources stored by the data provider. A service provider acquires metadata by sending HTTP requests to data providers. The harvested metadata records are stored and can be used to provide other services, e.g. paper recommender services. How the data obtained by a service provider is used is outside the scope of this protocol specification.

4.4.4 Requests and Responses

Xharvester supports two types of requests: *ListMetadataFormats* and *ListRecords*.
These requests are used with additional parameters to fine-tune the response(s). The parameters are: *requestType*, *resumptionToken*, *metadataFormat*, *from* and *until*. Table 4.2 show the types of requests and their use of parameters.

Type of Request	**Description**	**Parameter Use**
ListMetadataFormats	Lists all metadata formats supported by the repository	None
ListRecords	Lists the records available in the specified format	Mandatory parameters: metadataformat. Optional Parameters: from, until, resumptionToken.

Table 4.2: Xharvester Types of Request and Parameters use.

[8]http://www.loc.gov/marc/bibliographic/
[9]http://www.loc.gov/standards/mods/

4.4.4.1 Parameters

There are 6 parameters to be used with the requests, namely *requestType*, *metadataFormat*, *identifier*, *resumptionToken*, *from* and *until*. The parameters are either mandatory or optional, and one parameter may be mandatory for some requests and optional to others.

4.4.4.1.1 requestType The *requestType* parameter is used to specify the type of request being sent to the server, therefore it is a mandatory parameter used in all requests.
Request Example:
http://www.example.com/Xharvester.pl?requestType=ListMetadataFormats

4.4.4.1.2 metadataFormat The metadataFormat parameter is used to specify the format of the record(s) to be harvested. The use of this parameter is mandatory for the *ListRecords* request. The use of this parameter with any other request should generate an error.
The value of the metadataFormat parameter should be one that is indicated as supported by the ListMetadataFormats request.
Request Example:
http://www.example.com/Xharvester.pl?requestType=ListRecords&metadataFormat=dc

4.4.4.1.3 resumptionToken The *resumptionToken* is an optional parameter that is sent in the response when the server sends an incomplete response to a client request. The client can then use the resumption token to retrieve the outstanding records. resumptionToken is used with the ListRecords request. The use of this parameter with any other request should generate an error.
How many records are sent per response is defined by either server settings or by the use of the date parameters, from and until. For example, if a server is set to send 100 records at a time, but a request results in a response with 500 records, the server will sent 100 records and a *resumptionToken* until all records that match the request have been sent. In this case it will send 5 response pages, the first 4 with resumption tokens and the last one either without a resumption token or with an empty resumption token field, to indicate that there are no more records left.
The format of the *resumptionToken* is defined by the sever. It can be encoded (resumptionToken=487dfs9) or explicit (resumptionToken=100-200).
Request Example:
http://www.example.com/Xharvester.pl?requestType=ListRecords&metadataFormat=dc
&resumptionToken=346985c5gvc
An expiryDate value can optionally be sent with the resumptionToken to be used for the ListRecords request. This states a date at which the resumptionToken loses its validity. It may be the date after which the repository contents will be changed (adding, editing and deleting content). The use of the resumptionToken after the expiryDate **SHOULD** generate an error. The dates used for expiryDate, from and until **SHOULD** follow theW3C[10] Date and Time Formats for Coordinated Universal Time[11].

[10]http://www.w3.org/standards/
[11]http://www.w3.org/TR/NOTE-datetime

4.4.4.1.4 from and until The *from* and *until* parameters are optional date parameters used for selective harvesting. They allow the harvester to specify the time frame of the records to be harvested. These parameters are used with the *ListRecords* request. When *from* and *until* are not used, the server sends a response that includes records from the earliest to the latest dates recorded in the server.
Request Example: http://www.example.com/Xharvester.pl?requestType=ListRecords &metadataFormat=dc&from=2011-01-01&until=2011-09-07

4.4.4.2 Types of Requests

4.4.4.2.1 ListMetadataFormats *ListMetadataFormats* queries the server about the metadata formats it supports. It would normally be the first request to sent, since all other requests require the use of the *metadataFormat* parameter.
Example Request:
http://www.example.com/Xharvester.pl?requestType=ListMetadataFormats
Example Response:
<HIP-CFxharvester>
<request>ListMetadataFormat</request>
<metadataFormat>dc</metadataFormat>
<metadataFormat>mets</metadataFormat>
</HIP-CFxharvester>

4.4.4.2.2 ListRecords *ListRecords* provides a list of records in a specified format. It uses mandatory parameter *metadataFormat* and one or more optional parameters as specified in Table 4.2.
Example Request: http://www.example.com/Xharvester.pl?requestType=ListRecords &metadataFormat=dc
Response:
<HIP-CFxharvester>
<request>ListRecords</request>
<resumptionToken>1skgaf34702</resumptionToken>
<expiryDate>2011-09-07</expiryDate>
<record>
<identifier>000038XYZ</identifier>
<contributor>Jane Doe</contributor>
<creator>Jane Doe</creator>
<creator>John Doe</creator>
<date>2000-03-25</date>
<language>English</language>
<title>Example Resource File</title>
<description>This is an example of an XML file in the Dublin Core metadata format... Dublin Core has 15 elements, it is very flexible because of optionality and repeatability
</description>
<relation>http://www.example.com/000038XYZ</relation>
<type>Monograph</type>
</record>

<record>
<identifier>000487XYZ</identifier>
<contributor>Mary X</contributor>
<creator>Jane Paul</creator>
<date>2003-05-22</date>
<language>Portuguese</language>
<title>Examplo de um ficheiro</title>
<description>
Este e um exemplo de um ficheiro em XML, no formato de metadados Dublin Core. Esse formato tem 15 elementos. E oferece flexibilidade por muitos serem opcionais e repetiveis
</description>
<relation>http://www.example.com/000482XYZ</relation>
<type>Monografia</type>
</record>
</HIP-CFxharvester>

4.4.4.3 Error Messages

The server will generate an error if it is unable to decode the request sent. The server will not be able to decode a bad request. A request is considered a bad request if the *requestType* and/or any of the other required parameters is misspelled, do not exist or are missing; and if the values for the parameters are not in the correct format or are missing. It is advisable that the server provides the most informative possible error messages.

4.5 Xbrowse

4.5.1 Introduction

Browsing is the process of going through a collection of items using specific criteria to find the items of interest.
Xbrowse protocol is a high-level interoperability protocol that provides a simple service for clients to browse through collections of resources from digital libraries, databases and other sources of digital data.

4.5.1.1 Protocol Model

The protocol has two main components: a data repository or a server, a client application.

4.5.1.1.1 The Data Repository The data repository is where the collection of resources is stored. They can be developed by the Xbrowse implementer or belong to another institution or individual. As long as the repository gives the client application the necessary privileges for data access, ownership is not an issue. The Xbrowse implementer **MUST** ensure that they have access to the repository records and know the structure of the repository (indexes).

Classifiers are categories used to index resources in repositories. They allow records with similar attributes to be displayed together when clients send requests for resources in a specific classification group.
The client application can provide browsing support for as many of the repository classifiers as wanted - it is not mandatory to use all the classifiers the repository provides. The classifiers can be used individually or combined, for example browse by type and year; and a classifier may have inner classification options, for example classifier age may have inner classifiers 0-10, 11-20, etc.
A Xbrowse implementation may be developed to browse one or multiple repositories.

4.5.1.1.2 The Client Application The client uses this protocol to browse the collection(s). The browsing is done by choosing one or more classifiers. Classifier examples are: Year, Author, Status, Country, etc.
When the client application sends a request to the server, the server processes the request and then returns a response containing records that are categorised by the chosen classifier(s). The results page can contain all records found, however it is **RECOMMENDED** that in case of a large number of results, subjective to the developers choice and size of each record, that the results are displayed at a limited number of records at the time. If no records indexed by the classifier are found the client **SHOULD** receive a no match found message.
The choice of a classifier denotes that the records returned will all be linked by a similar attribute. For example a classifier for Language will provide a list of the different language options available in the repository; and the choice of a language, for example Portuguese, will display all the resources in the repository that have been written or have any connection to the language of choice.

4.5.2 Browsing Examples

4.5.2.1 General Browsing - Request

When a general browsing request is sent to the server, the response is an XML file that contains a list of the records available in the server. That list is not necessarily organised by any specific criteria. The server response **SHOULD** also include a list of the classifiers it supports. A general request could have a format similar to http://174.19.284.1/gina/cgi-bin/browse.pl
General Browsing - Response
<Xbrowse>
<classifier>year</classifier>
<classifier>status</classifier>
<classifier>type</classifier>
<record>
<title>Integrated Query of the Hidden Web </title>
<author>Berman Sonia </author>
<author>Kamkuemah Martha </author>
<author>Muntunemuine John </author>
<description>There is a need for software that can access multiple Websites through a

single, common interface. This would allow users, for example, to compare flights for a particular trip across all relevant airline sites by posing a single query. This paper investigates automating this process in the case of airline databases hidden behind the Web (the so-called Deep Web or Hidden Web). We first constructed a prototype for integrated query of a handful of pre-determined airline sites. This proved useful in detecting commonalities and differences in the sites, and in selecting the most suitable technologies for working with multiple forms. A generic system was then designed and components of the prototype incrementally replaced by domain-specific tools able to handle arbitrary airline sites. Our results were promising as regards result interpretation, with 89% of response pages successfully handled. However query formulation presented many problems, with only 39% of query interfaces automatically interpreted correctly, and even fewer amenable to automated query propagation. We conclude that integrated access to the Hidden Web is considerably more challenging than crawling the Surface Web, and that domain-specific systems are a promising approach to full automation.</description>
<year>2010</year>
<type>Conference or Workshop Item</type>
<status>Published</status>
</record>
<record>
<title>Visual Dictionary for an Extinct Language, A </title>
<author>Williams Kyle </author>
<author>Manilal Sanvir </author>
<author>Molwantoa Lebogang </author>
<author>Suleman Hussein </author>
<description>Cultural heritage artefacts are often digitised in order to allow for them to be easily accessed by researchers and scholars.</description>
<year>2010</year>
<type>Conference or Workshop Item</type>
<status>Submitted</status>
</record>
</Xbrowse>

4.5.2.2 Browse By Classifier - Request

When browsing by a classifier, for example browsing by status, the request should also include the classifier parameter value and if necessary one or more inner classifier parameter options. In this example, http://174.19.284.1/gina/cgi-bin/browse.pl?classifier=status&status=unpublished.

Browse By Classifier- Response

<Xbrowse>
<classifier>status</classifier>
<classification>Unpublished</classification>
<record>
<title>MsIP</title>
<author>Paihama Jorgina Kaumbe do Rosario </author>
<description>Interoperability can be defined as the ability of different entities to com-

municate with each other. In a Digital Libraries context this would involve a system X being able to search and/or retrieve data from system Y. Due to the increasing nature of people building systems that are connected via the Web, it is important to provide the simplest and more efficient possible means for systems to interoperate. This paper presents a proposed study on Meta-standardisation of Interoperability Protocols. The aim of this study is to first review the current state of interoperability between data storage systems provided by communication protocols; then design an interoperability framework that can possibly improve on the current set of protocols. </description>
<year>2009</year>
<type>Conference or Workshop Item</type>
<status>Unpublished</status>
</record>
</Xbrowse>

As a proof of concept the three services were implemented. More details on the implementation are given in Chapter 5.

Chapter 5

Evaluation and Analysis

An evaluation was done in order to determine if the research done answers the research question. In this case does this research answer the question: "Is it possible to develop a uniform suite of simple and efficient interoperability protocols to improve on the current medley of protocols?" The evaluation results prove or, in same cases, disprove the research question.
Three distinct methods of evaluation were used:
1. Case Study - implementation of the protocol services.
2. User Understandability Evaluation - a study of how well users understand protocols.
3. Entropy Calculations - calculations of the information content of XML files.
The sections below describe how different metrics are used to evaluate if and how this experimental protocol answer the research question.

5.1 Eprints Case Study

The first step of the evaluation was a case study of the practicality of implementing the suggested experimental protocol.
An EPrints digital library was used as the server for the case study. Figure 5.1 shows the EPrints user interface. Although the user interface was available, for protocol requests the client application sends HTTP requests via the URL and not by using the HTML forms available on the user interface. All response files are XML files. XML is the best format for inter systems communications because it facilitates the client's understanding and usage options for the data obtained.
The client applications were all developed in Perl and the query language used was MySQL version 5.1.54. EPrints has available various API's to help programmers create plug-ins or any value added services they may need/want to support.

Figure 5.1: EPrints Repository User Interface

The case study involved the implementation of all all proposed services, namely search, browse and harvest. All services were successfully implemented and each protocol was able to provide the services as expected. Figures 5.2 and 5.3 show the use of the Xsearch and Xbrowse protocols respectively, a result from Xharvester can be seen in Figure 5.16.

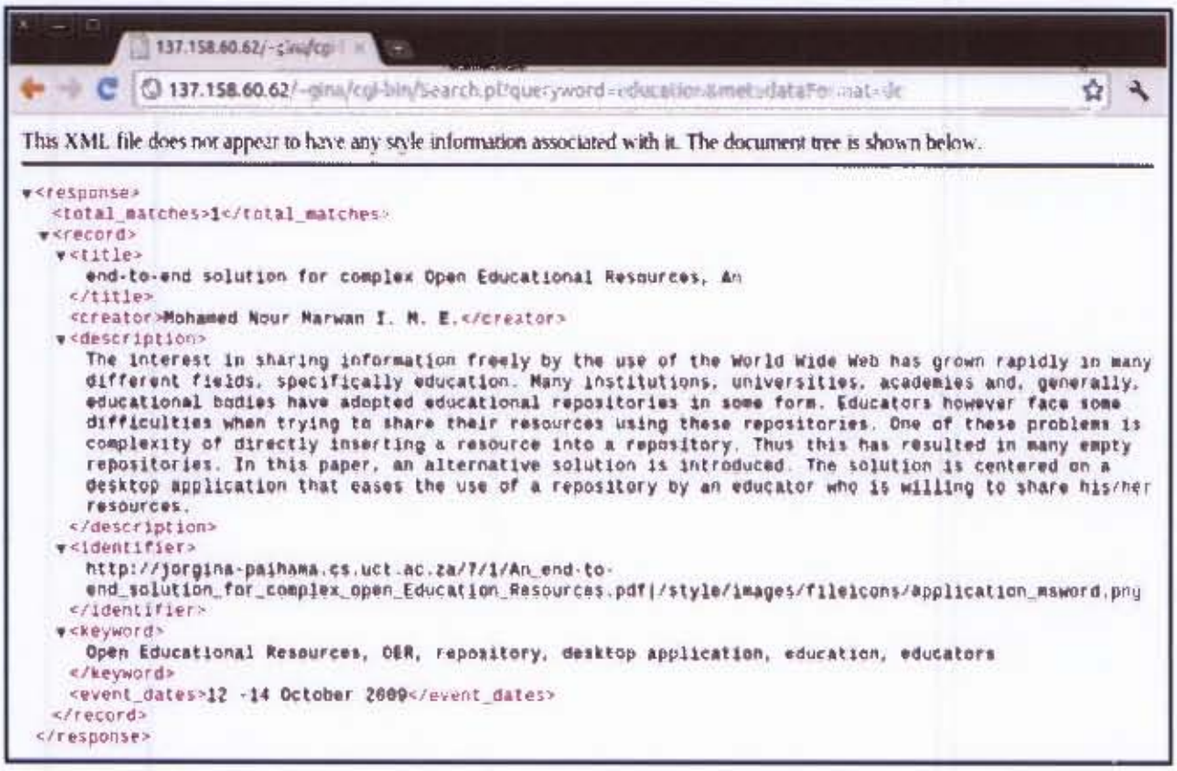

Figure 5.2: Xsearch Protocol Result

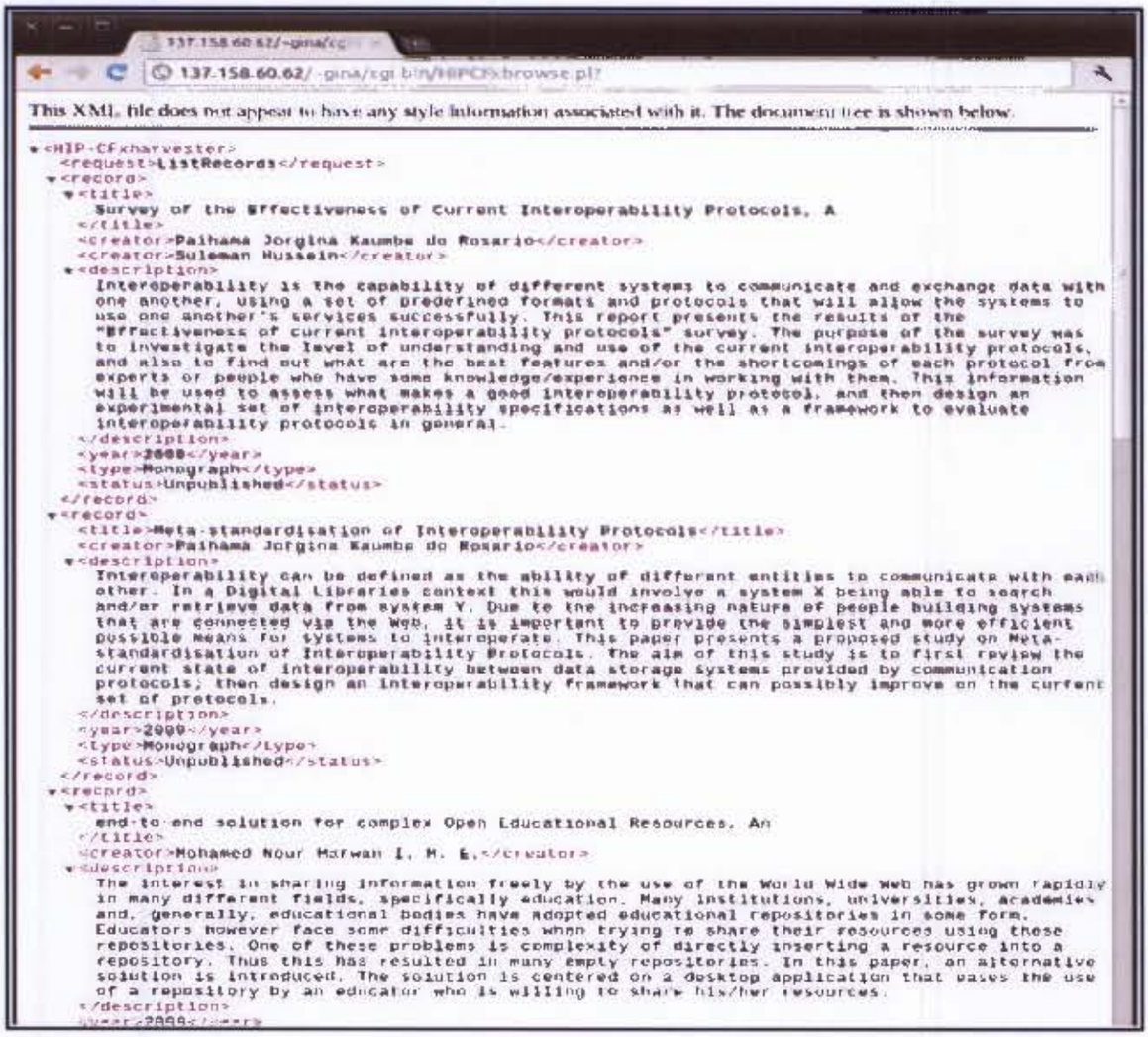

Figure 5.3: Xbrowse Protocol Result

5.2 Complexity

Based on the results from the user survey (see chapter 3) the HIP-CF suite of services was designed to be efficient and reliable but, most importantly, be simple, as opposed to complex and computationally expensive. Simplicity is assumed be a key factor to reduce costs, reduce the usage of resources both human and material, save time and possibly increase adherence of users.
There are a number of possible ways to evaluate complexity/simplicity:

Complexity Theory Evaluation

Adapting the computational complexity theory definition[1] to fit this research, a complex protocol is defined as a protocol that solves a problem for which a simpler protocol can be used.
For each given problem one specific approach/protocol may be simpler and/or be a better solution than another. This is what this evaluation will attempt to demonstrate. There is no one solution that is the "best" for all problems - finding the best solution is as subjective as the needs and knowledge of the implementers.

5.2.1 Understandability Complexity

The understandability complexity experiment is a measure of how complex or simple it is to understand the protocol. Users were given the documentation of different protocols, one of them being the documentation of HIP-CF and the other(s) being the documentation of a protocol or protocols that provides equivalent services. Based on their understandings they completed a questionnaire to indicate what they thought was the simplest protocol.
Note: this was not an evaluation of the individual features supported by the protocols, i.e. users did not evaluate how a protocol supports search or browse or resumption tokens or sets, etc, to then compare it to an equivalent protocol. In fact a generalised approach to what the protocol does was taken, rather than evaluating each individual feature supported. The functionality and features of SRU and OAI-PMH are covered in chapter 2 and those of HIP-CF in chapter 4.

5.2.1.1 Pre User Evaluation Work:

In order to avoid any bias towards or against the HIP-CF, when compared to the existing well established standards, the documentation was given to 5 masters students in the digital libraries laboratory at the Computer Science department at the University of Cape Town. In a focus group, the students were asked to read the documentation of the protocol and write comments, ask questions, give suggestions and their overall opinion about the documentation. They were asked to comment on grammar, writing and presentation style and content. Since this study aims at simplifying solutions available for high-level interoperability, the objective here was to get an outsider's perspective and ensure that the HIP-CF documentation is not written in a way that over simplifies things in order

[1] http://en.wikipedia.org/wiki/Computational_complexity_theory

to attract the favouritism of participants in the user study. Changes were made to the documentation based on the feedback from the focus group.
The feedback from the focus group led to a reduction in the size of the documentation as a result of even more simplification of the protocol services, but no major changes were made. The focus group was also important in determining the amount of time necessary for the user experiments.
Each user only participated in the evaluation once, i.e. users who participated in the pre-evaluation work did not participate in the actual evaluation and vice-versa.
The evaluation involved a total of 27 people plus 5 people in the pre-evaluation work. All participants are Computer Science students.

5.2.1.2 Individual Protocol Service Evaluation

The individual protocol service understandability evaluation involved a total of 23 participants. The sessions were conducted in a controlled experiments environment.
The participants were invited to be part of the evaluation session via emails through mailing lists, flyers and word of mouth. Each session lasted between 01:00 to 01:30 hours. Sessions started with a brief explanation of the project and an explanation of how the evaluation session was going to be carried out.
Each user was given the documentation of two protocols to read. One was a HIP-CF service (preamble + specific service documentation) and the other was the documentation of a protocol that provides an equivalent service, i.e. a user that received Xsearch also received the documentation for SRU, both of them being protocols for search and retrieval of resources, and Xharvester was compared to OAI-PMH, both of which are data harvesting protocols. To the best of my knowledge there is no formal protocol specification for a high-level browsing protocol, therefore Xbrowse was not part of the understandability evaluation.
Additionally, participants were also given a consent form (see Appendix D) and a questionnaire (see appendix E1). The users were asked to read and sign the consent form before they could begin the reading process.
After reading both documents the users were asked to complete the questionnaire.

Individual Protocol Service Evaluation Results: Out of the total of 23 participations, 6 were null answers, 10 responses provided a comparison between Xsearch and SRU, and 7 responses provided a comparison between Xharvester and OAI-PMH.

5.2.1.2.1 Null Answers The answers by 6 participants were considered null for the following reasons:

1. 4 participants compared the HIP-CF preamble to its own services: Xsearch or Xharvester.

   (a) Possible reasons for why some participants compared the HIP-CF preamble to its own services, i.e. Xsearch or Xharvester, instead of Xsearch vs. SRU or Xharvester vs. OAI-PMH:

i. At the beginning of the session users where given an explanation that stated the fact that HIP-CF and Xsearch or Xharvester were two parts of the same framework (the preamble and the actual protocol service) and that SRU or OAI-PMH was the protocol to compare it with. Based on the fact that such explanation was provided and the fact that the other users understood and compared the correct protocols to each other, it is possible that this was motivated by the nature of the evaluation. The 4 participants could have been distracted or read at a slower pace, when compared to the rest of the group, and therefore used the information from what they had managed to read when answering the questionnaire.

ii. The participants who got bored only read one of the protocols.

iii. Due to the fact that they did not understand the first protocol, the participants did not even try to read the second one.

iv. The researcher was not clear on the explanations about the evaluation procedure and the purpose of the research.

v. The participants misunderstood the researchers' explanations.

2. 1 participant understood and named the two different protocols to be compared, but his/her answers to the questionnaire were unusable because he/she clearly stated in question 9 that he/she did not understand either one of the protocols. That was also clear from the fact that the participant wrote that both Xsearch and SRU provide search, harvest and browsing services.

3. 1 participant compared Xsearch to HTTP.

5.2.1.2.2 Xsearch vs. SRU A total of 10 participants compared Xsearch to SRU. Out of the 10 participants, 2 were honours students and 8 were undergraduate students. Based on their answers, the following results were obtained.

Programming Skills: The 10 students described their programming skills as follows:

- 2 participants said that they are expert programmers.
- 4 participants said that they are good programmers.
- 4 participants said that they are average programmers.

Findings: 40% of the programmers were average and 60% was above average.That makes them qualified to assess the possible degree of difficulty involved in implementing a protocol.

Previous knowledge of high-level interoperability protocols: When asked if they knew any high-level interoperability protocols prior to this evaluation, the participants responded as follows:

1. 7 no
2. 2 yes

3. 1 null (yes & no)

Findings: 70% of the participants said that they had no prior knowledge about high-level interoperability protocols. While not all high-level, the protocols named by the participants with previous knowledge were: routing protocols OSPF (Open Shortest Path First) and RIP (Routing Information Protocol); and application layer protocol HTTP. One user named Google as a protocol.

Level of Understanding Ratings: The participants were asked to compare their level of understanding of the two protocols, on a scale of 1 to 10 where 1 is 'I still don't understand the protocol' and 10 is 'I now have a good understanding of the protocol'. Figure 5.4 shows the results for the levels of understanding ratings of both Xsearch and SRU.

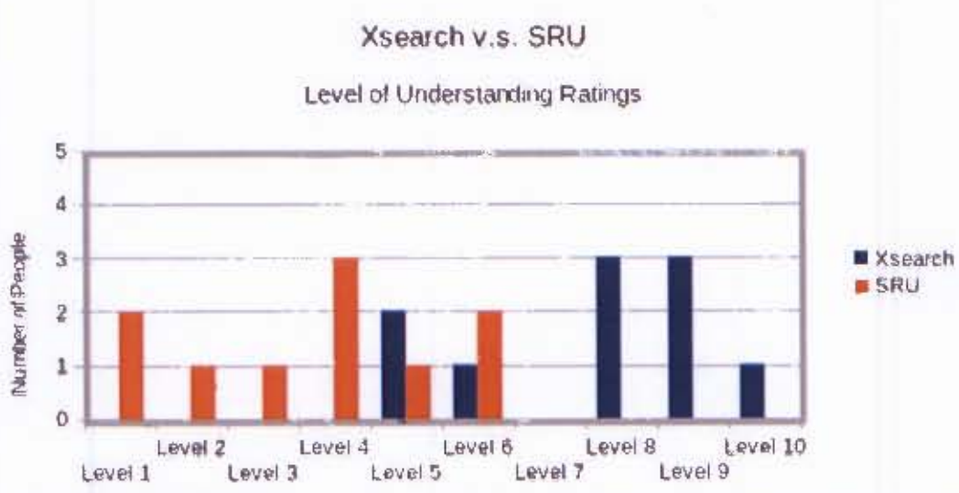

Figure 5.4: Xsearch vs. SRU Levels of Understanding Ratings

Findings: Xsearch - 80% of participants rated their levels of understanding of Xsearch above 5 which is the average level of understanding, and 20% chose level 5.
With 70 % above average and no values under the average line, the results indicate that Xsearch is very simple to understand.
Findings: SRU - 80% of participants rated their levels of understanding of SRU below or at level 5, and 20% chose level 6.
With 70% below average the results indicate that SRU is not very simple to understand.
The *arithmetic mean* level of understanding for Xsearch is at level 7.7 (closer to 'I now have a good understanding of the protocol') and 3.6 for SRU (closer to 'I still don't understand the protocol'). Comparing Xsearch to SRU in terms of levels of understanding gives us 80% above average for Xsearch and 70% below average for SRU. Based on this result, Xsearch is potentially simpler to understand than SRU.

Writing and Presentation Style Ratings: The participants were asked to compare the writing and presentation styles of the two protocols, on a scale of 1 to 10 where 1 is 'poor' and 10 is 'very good'. Figure 5.5 shows the results of the comparison of writing and presentation style ratings for Xsearch and SRU.

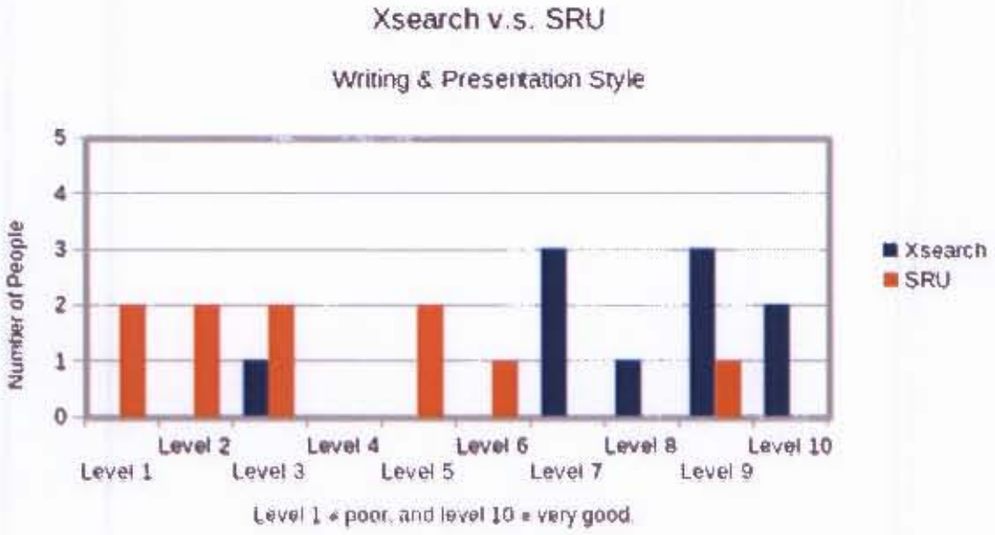

Figure 5.5: Xsearch vs. SRU Writing & Presentation Style Ratings

Findings: Xsearch - 90% of participants rated the writing and presentation style between levels 7 to 10. 10% chose level 3.
With 90% of participants rating this category above the average line, the results indicate that Xsearch's writing and presentation style is good and simple for users to follow and comprehend.
Findings: SRU - 80% of participants rated the writing and presentation at or below the average line, and 20% rated it at or above level 6.
With 80% of participants rating this category at average or below average, the results indicate that SRU's writing and presentation is not simple to follow and comprehend.
The arithmetic mean for the writing and presentation style of Xsearch is 7.9 (closer to very good) and 3.7 for SRU (closer to poor). Comparing Xsearch to SRU in terms of writing and and presentation style the results shows us that the majority of users (90%) are above average, which indicates that they are satisfied with Xsearch's writing and presentation style and also the majority of users (60%) are below the average line for SRU, which indicates that they are not satisfied with SRU's writing and presentation syle . Based on this result, Xsearch potentially simpler than SRU.

Perceived Degree of Implementation Difficulty: The participants were asked to rate how difficult they think the implementation of the protocols would be, on a scale of 1 to 10 where 1 is 'easy' and 10 is 'difficult'. Figure 5.6 shows the results of the perceived degree of difficulty associated with the implementations of both Xsearch and SRU.

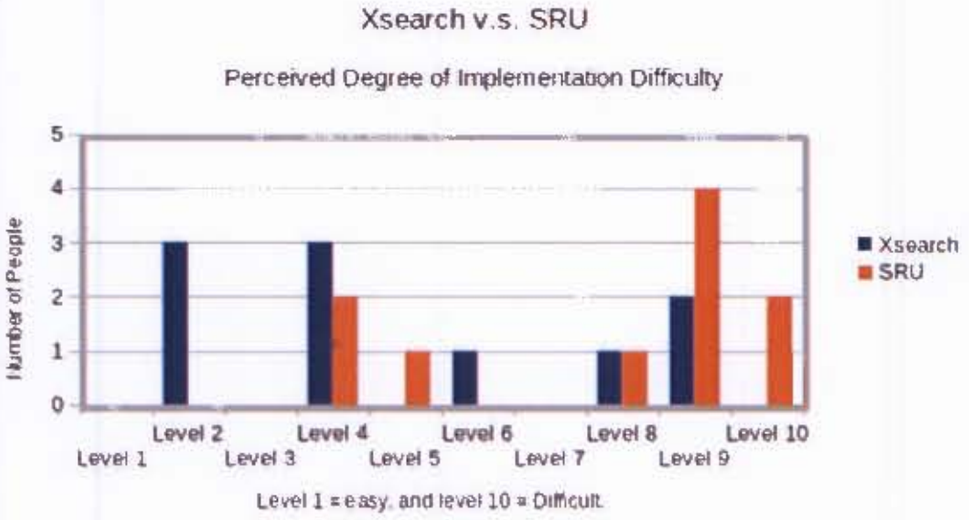

Figure 5.6: Xsearch vs. SRU Degree of Implementation Difficulty Ratings

Findings: Xsearch - 60% of participants rated their perceived degree of difficulty associated with implementing Xsearch to be below level 5, which is from easy to just below average, and 40% of participants rated this category between level 6 and level 9, which is between average and difficult.
The results indicate that even some of the people who find Xsearch easy to understand and rated the writing and presentation style as good, did not perceive its implementation to be easy.
Findings: SRU - 30% of participants rated their perceived degree of difficulty associated with implementing Xsearch to be at or below level 5, which is from easy to average, and 70% of participants rated this category between level 8 and level 10, which is from somewhat difficult to difficult.
The number of participants that perceive an implementation of SRU to be from above average to difficult is consistent with the number of participants that did not find SRU easy to understand and rated its writing and presentation style below average. From the results it can be assumed that the perceived degree of implementation difficulty is directly related to the participants' difficulty of understanding the protocol.
The arithmetic mean for the perceived degree of difficulty associated with Xsearch's implementation is 5 (average) and 7.7 for SRU. Comparing Xsearch to SRU in terms of perceived degree of difficulty associated with the implementations, the results indicate that although the number is higher for SRU, for both protocols more than 50% of participants don't perceive implementation to be a trivial task. However, based on the mean values, Xsearch's implementation is expected to be easier than that of SRU.

Correlation between participants' programming skills and their responses to the questionnaire: Table 5.1 show the participants' self described level of programming skills and their choices for each main comparison category.

Programming Level	Q6 Xsearch	Q6 SRU	Q7 Xsearch	Q7 SRU	Q8 Xsearch	Q8 SRU
Expert 1	10	4	10	2	2	9
Expert 2	9	1	10	1	2	10
Good 1	9	6	7	9	9	5
Good 2	5	1	3	1	6	10
Good 3	9	4	9	6	9	4
Good 4	6	3	9	5	8	4
Average 1	8	5	8	3	4	9
Average 2	8	4	7	5	4	8
Average 3	5	6	7	3	4	9
Average 4	8	2	9	2	2	9

Table 5.1: Participants' programming skills and their answers

Correlation:

- Question 6 - Understandability:
  - With a difference of just one level (9 to 10) it is possible to say that both *expert programmers* agree on the level of understanding of Xsearch, and although their levels of understanding of SRU are not very close (1 and 4) both are below average. Note that none of them had any previous knowledge of high-level interoperability protocols.
  - Except for one participant who rated Xsearch at level 6, none of the other three *good programmers* had any previous knowledge of high-level interoperability protocols. Two out of three rated Xsearch at level 9 while the other participant chose level 5. The understandability of SRU was also rated the highest by those who rated Xsearch the highest although at slightly lower level (i.e. Xsearch 9 - SRU 6, Xsearch 9 - SRU 4, Xsearch 6 - SRU 3, Xsearch 5 - SRU 1).
  - Three of the *average programmers* understood Xsearch at level 8 and the other one at level 5, while for SRU it was two participants below the average line, at levels 2 and 4, one participant at level 5 and one at level 6. Two of the average programmers had previous knowledge of high-level interoperability protocols (the ones who rated Xsearch 8 - SRU 5, and Xsearch 8- SRU 4) and two did not. Note that the participant who rated Xsearch the lowest (5), is the one who rated SRU the highest (6).
    * The level of programming skills does not seem to have a big impact on the levels of understandability. This is based on the fact that although Xsearch was rated highest by the people with the highest programming skills, SRU was better understood by the average programmers than it was by the good programmers, and it was better understood by the good programmers than by the expert programmers. Therefore this leads to

the conclusion that understandability is influenced by other factors more than it is influenced by the level of programming skills.

- Question 7 - Writing and presentation style:
  - The *expert programmers* both rated Xsearch's writing and presentation style very high at level 10, and SRU very low at levels 1 and 2. Note that as with understandability there is a large difference between the ratings for each protocol.
  - In the *good programmers* group the one participant who gave both protocols the lowest ratings for understandability (Xsearch 5 - SRU 1), also gave them the lowest ratings for writing and presentation style (Xsearch 3 and SRU 1) suggesting that he/she did not really understand and was also not satisfied with the writing and presentation style of either one of the protocols. The other 3 participants gave both protocols scores above the average line, i.e Xsearch 7 - SRU 9, Xsearch 9 - SRU 6, Xsearch 9 - SRU 5. It is interesting to note that one of the participants who rated Xsearch at level 9 and SRU at level 6, in terms of understandability, which would imply that Xsearch is relatively easier to understand, then rated SRU's presentation style at level 9, and Xsearch at level 7, suggesting that he/she found that Xsearch's writing and presentation style is not better than SRU's although it is simpler to understand.
  - All four *average programmers* rated Xsearch's writing and presentation style between levels 7 to 9, and SRU was rated at level 2 by one participant, two participants at level 3 and one participant at level 5.
    * For Xsearch the overall highest ratings where given by the expert programmers, followed by the average programmers and the good programmers had the overall lowest ratings. For SRU the highest ratings were given by the good programmers, followed by the average programmers and the expert programmers had the overall lowest ratings.
- Question 8 - Implementation difficulty:
  - Both *expert programmers* rated Xsearch quite easy to implement, at level 2, and SRU quite difficult to implement at levels 9 and 10. The difference between the values for each protocol is as noticeable as in the other two categories but the interesting fact to note here is that although both participants are self-described expert programmers they both consider SRU difficult to implement. That indicates that their perceived implementation difficulty was more influenced by understandability than by programming skills.
  - For the *good programmers* Xsearch's perceived implementation difficulty is also closely related to understandability, but the fact to note here is that the people who claimed to better understand Xsearch were the ones who perceive its implementation to be the hardest. Also to note that overall the participants rated Xsearch as being simpler to understand with a good writing and presentation style, but all participants rated SRU as being easier to implement. Unlike the expert programmers, understandability is not directly related to implementation difficulty for the good programmers.

- Much like the experts, *average programmers* also rated their perceived difficulty closely related to understandability, with Xsearch as being easier (below average) and SRU as being more difficult to implement, at levels 8 and 9.
    * The level of programming skills was expected to have the biggest and most predictable impact on this issue, but this was not the case. Except to the average programmers, for all participants, the perceived degree of implementation difficulty was influenced by their levels of understanding. Programming skills and previous knowledge of high-level interoperability protocols were not very strong influencing factors. When the implementations were rated as difficult (Xsearch for good programmers, and SRU for expert and average programmers) the levels of difficulty were high regardless of the programming skills (i.e. experts had an arithmetic mean of 9.5 (= (9+10)/2), the good programmers had an arithmetic mean of 8 (= (9+6+9+8)/4) and the average programmers 8.75 (= (9+8+9+9)/4). The arithmetic mean for difficulty was actually higher for the expert programmers. This finding suggest that it may be better to present "not so good" programmers with "good" protocols than it is to get "good" programmers to implement "not so good" protocols.

General comments: The last question asked participants to make comments about the protocols. See the comments followed by a brief discussion below.
Xsearch

1. Simple and easy to use.
2. Cost effective (computationally).
3. Not very secure.
4. Possibly slow data access.
5. Should include a diagnostic model.
6. Ideal for new users.

SRU

1. Very long and complex documentation.
2. Computationally expensive.
3. Robust and reusable.
4. Hard to understand and implement.
5. Good, precise and provides diagnostic support for errors.
6. Goes too deep into lower levels descriptions.
7. Too many acronyms, hard to follow.

8. Requires previous knowledge of many different standards.
9. Needs practical and use case examples.
10. Variety of parameters makes it adaptable to many systems.
11. Only understood what it does towards the end.
12. Have a brief and summarised version for quick referencing.

Xsearch vs. SRU Conclusion
The overall conclusion from the comparison of Xsearch and SRU is that Xsearch is not only simpler to understand but also has a more usable writing and presentation style than SRU. Although the ratings of the perceived degree of implementation difficulty between the two protocols are very close to each other, from this result it is possible to assume that because of the advantage in the other categories Xsearch is likely to have higher adherence, which supports the claims made by this research and imply that the SRU protocol could benefit from greater simplicity.

5.2.1.2.3 Xharvester vs. OAI-PMH A total of 7 participants compared Xharvester and OAI-PMH. Based on their answers, the following results were obtained.

Programming Skills: The 7 participants described their programming skills as follows:

- 3 participants said that they are good programmers.
- 4 of participants said that they are average programmers.

Previous knowledge of high-level interoperability protocols: When asked if they knew any high-level interoperability protocols prior to this evaluation, all 7 participants responded no. All participants were introduced to high-level interoperability for the first time at the evaluation session.

Level of Understanding Ratings: The participants were asked to compared their level of understanding of the two protocols, on a scale of 1 to 10 where 1 is 'I still don't understand the protocol' and 10 is 'I now have a good understanding of the protocol'. Figure 5.7 shows the results for the levels of understanding ratings of both Xharvester and OAI-PMH.
Findings: Xharvester - All participants rated their levels of understanding of Xharvester above average, all between levels 7 and 9.
With all ratings above average, the results indicate that Xharvester is very simple to understand. The result is relatively similar to that of its "sibling" protocol Xsearch, where 80% of participants also rated their level of understanding above average.
Findings: OAI -PMH 86% of participants rated their levels of understanding as average or below average, and the other 14% chose level 8.
With over 80% of the ratings below average the results indicate that OAI-PMH is not very simple to understand. The interesting fact here is the large difference between the lowest rating (level 3) and the highest rating (level 8).

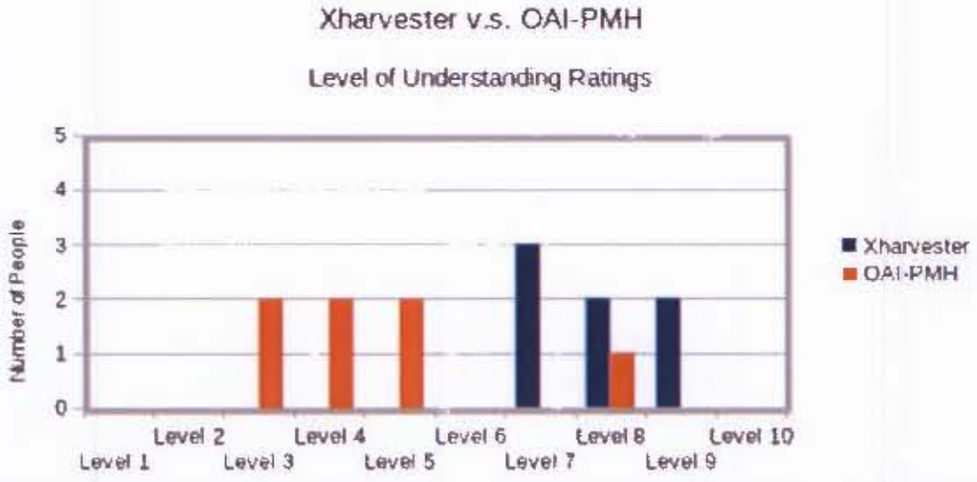

Figure 5.7: Xharvester vs. OAI-PMH Levels of Understanding Ratings

The arithmetic mean level of understanding for Xharvester is at level 7.9 (closer to I now have a good understanding of the protocol) and 4.6 for OAI-PMH (closer to I still dont understand the protocol). Comparing Xharvester to OAI-PMH in terms of levels of understanding, the numbers clearly show that Xharvester, with 90% above average is simpler to understand than OAI-PMH with 60% below average. Based on the arithmetic mean values Xharvester is potentially simpler to understand than OAI-PMH.

Writing and Presentation Style Ratings: the participants were asked to compare the writing and presentation styles of the two protocols, on a scale of 1 to 10 where 1 is 'poor'and 10 is 'very good'. Figure 5.8 shows the results for the writing and presentation style ratings of both Xsearch and SRU.
Findings: Xharvester - All participants rated their levels of understanding of Xharvester above average, all between level 6 and 10, with level 8 having the most votes (3) and being the only one chosen by more than one participant.
The results indicate that, with all participants rating it above average, Xharvester's writing and presentation style is good, and simple to follow and comprehend .
Findings: OAI-PMH - 57% of participants rated the writing and presentation of OAI-PMH below average and the other 43% rated it average or above average.
With more than 50% of participants rating this category below average, the results indicate that OAI-PMH's writing and presentation can still be made simpler to facilitate readers/users' ability to follow and comprehend.
The arithmetic mean for writing and presentation style for Xharvester is 8.3 (closer to very good) and 4.4 for OAI-PMH (closer to poor). Comparing Xharvester to OAI-PMH in terms of writing and and presentation style the results show that Xharvester is potentially simpler.

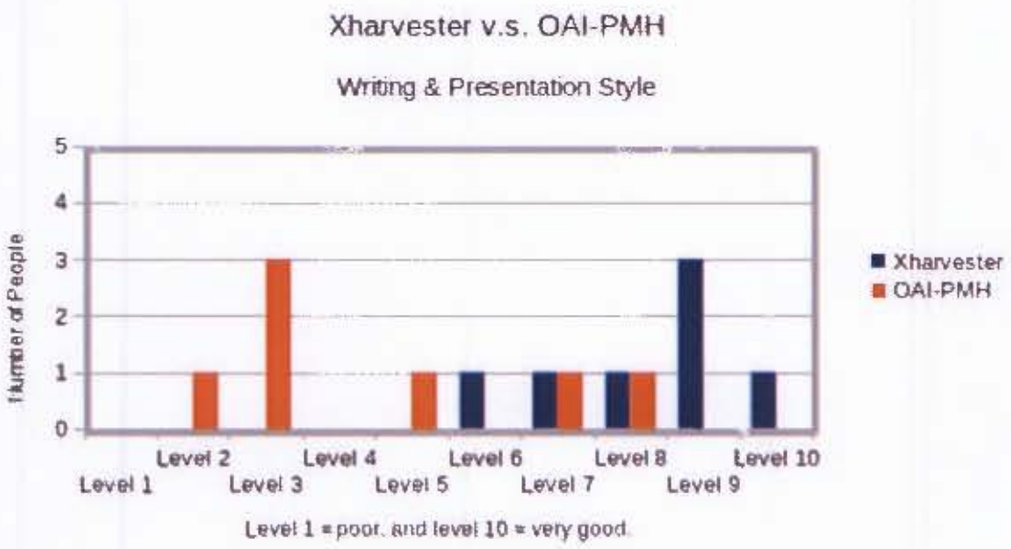

Figure 5.8: Xsearch vs. OAI-PMH Writing and Presentation Style Ratings

Perceived Degree of Implementation Difficulty: The participants were asked to rate how difficult they think the implementation of the protocols would be, on a scale of 1 to 10 where 1 is 'easy' and 10 is 'difficult'. Figure 5.9 shows the results of the perceived degree of difficulty associated with the implementations of both Xharvester and OAI-PMH.

Findings: Xharvester - 43% of participants rated their perceived degree of difficulty associated with implementing Xharvester to be below average, and the majority, specifically 57%, rated the implementation of Xharvester as above average difficult.

It is interesting to note that although Xharvester was rated as simple to understand and as having a good writing and presentation style by all participants(no ratings below the average line for both categories), more than 50% of participants do not see its implementation as an easy task.

Findings: OAI-PMH - All participants rated their perceived degree of difficulty associated with implementing OAI-PMH average or above the average line.

The results indicate that, by unanimity, participants agree that the perceived degree of difficulty associated with implementing OAI-PMH ranges from average to difficult. These results are not very surprising if the results from the level of understanding evaluation are taken into consideration.

The arithmetic mean perceived degree of difficulty associated with Xharvester's implementation is 5.7 (half way between easy and difficult) and 7.9 for OAI-PMH (closer to difficult). Comparing Xharvester to OAI-PMH in terms of perceived degree of difficulty associated with the implementations, the results show us that while the numbers differ for each protocol, the majority of participants see the implementation of either of the protocols to be a relatively difficult task. While this was expected for OAI-PMH based on the results from the previous categories, it is harder to understand these results for

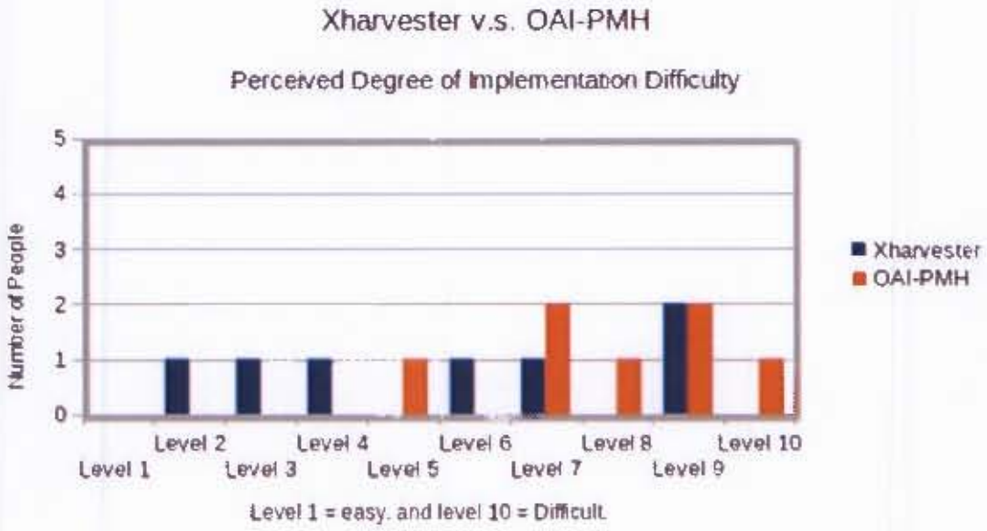

Figure 5.9: Xsearch vs. OAI-PMH Degree of implementation Difficulty Ratings

Xharvester.

Correlation between participants' programming skills and their responses to the questionnaire: Table 5.2 show the participants' self described level of programming skills and their choices for each main comparison category.

Programming Level	Q6 Xharvester	Q6 OAI-PMH	Q7 Xharvester	Q7 OAI-PMH	Q8 Xharvester	Q8 OAI-PMH
Good 1	9	8	6	8	4	7
Good 2	8	4	9	7	7	9
Good 3	7	3	9	2	2	10
Average 1	9	5	10	5	9	7
Average 2	7	4	8	3	6	8
Average 3	7	3	9	3	3	9
Average 4	8	5	7	3	9	5

Table 5.2: Participants' programming skills and their answers

Correlation:

- Question 6 - Understandability:
  - All three *good programmers* rated Xharvester above average from level 7 to 9, which indicates some agreement in their ratings of good understandability. For OAI-PMH, one of the participants rated it at level 8 and the other two participants rated it below average at levels 3 and 4, making the overall understandability by good programmers just average.

- The *average programmers* also rated Xharvester on the same range as the good programmers, from level 7 to 9. For OAI-PMH the ratings were between levels 3, 4 and 5, two below average and two average, taking the overall understandability of OAI-PMH to just below average.
    * Programming skills did not have too much of an impact here - both groups of programmers rated the protocols quite similarly. All programmers but one rated OAI-PMH between levels 3 and 5. Overall, good programmers gave Xharvester an arithmetic mean rating of 8 (= (9+8+7)/3) and OAI-PMH 5 (= (8+4+3)/3), and the average programmers rated Xharvester 7.75 (= (9+7+7+8)/4) and OAI-PMH 4.25 (= (5+4+3+5)/4). For both groups Xharvester is simpler to understand.

- Question 7 - Writing and presentation style:
    - Except for one participant's choice of level 2 for OAI-PMH, all *good programmers* rated the writing and presentation style of both protocols between levels 6 and 9, which means that overall they were satisfied with this category for both Xharvester and OAI-PMH.
    - While the *average programmers* all rated Xharvester above average, between levels 7 and 10, three of them rated OAI-PMH at level 3 and one at level 5. Contrary to the good programmers, they were not very satisfied with OAI-PMH's writing and presentation style.
        * The overall feeling was positive towards Xharvester's style, but the two groups disagreed when rating OAI-PMH. The good programmers liked the style but the average programmers did not like it.
          For good programmers the arithmetic mean for the understandability level of Xharvester is 8 (= (9+8+7)/3) and the arithmetic mean rating for style is also 8 (= (6+9+9)/3), and for OAI-PMH, understandability is 5 (=(8+4+3)/3) and style is 5.6 (= (8+7+2)/3).
          For average programmers Xsearch understandability is 7.75 (=(9+7+7+8)/4) and style is 8.5 (=(10+8+9+7)/4) and OAI-PMH understandability is 4.25 (=(5+4+3+5)/4) and style is 3.5 (=(5+3+3+3)/4), with only a small difference of 0.75 in each case. For both protocols the good programmers gave overall better ratings than the average programmers.

- Question 8 - Implementation difficulty:
    - The *good programmers* had very different perceptions of the degree of implementation difficulty for Xharvester (levels 2, 4 and 7), with an arithmetic mean of 4.3 (=(2+4+7)/3), which is more towards easy than difficult. For OAI-PMH all good programmers chose levels 7, 9 and 10, which makes an arithmetic mean of 8.6 (=(7+9+10)/3), more on the difficult side.
    - Except for one who chose level 3, all *average programmers* perceive the Xharvester implementation to be quite difficult (one at level 6 and two at level 9) at an arithmetic mean difficulty of 6.75, despite their claims on simplicity in understanding. OAI-PMH was also rated difficult at an arithmetic mean of 7.25.

* For the most part the good programmers were ahead of the average programmers in terms of finding things simpler and/or better in the different categories.In terms of the implementation difficulty of OAI-PMH,the average programmers, with an arithmetic mean of 7.25, think it is simpler to do it, when compared to the good programmers who rated this category an arithmetic mean of 8.6.

General comments: The last question asked participants to make comments about the protocols. See the comments followed by a brief discussion below.
Xharvester

1. Simplicity is impressive and may increase adherence.
2. Needs better error handling capabilities (handling exceptions).
3. More functionality to provide a wider range of options to users.
4. Needs to cater for data sets.
5. Simple and easy to understand, less is more.
6. Very well presented, tabular designs and diagrammatic presentation allows the user to understand the theory behind the protocol.
7. Xharvester could increase adherence but for the potential to be more successful than OAI-PMH lies the possibility of supporting all features supported by OAI-PMH while maintaining its simplistic approach.
8. Easier to understand, cover a minimal requirement set of features necessary for data harvesting.
9. Flexible data storage, user friendly, uses existing technologies.

OAI-PMH

1. The documentation is more extensive, and therefore initially seems that understanding and implementing may be harder but it the long run it seems to be easier to use and clearer than Xharvester.
2. Not flexible, communities have to adopt guidelines for sharing metadata prefixes.
3. Because it supports too many features the implementation can be expensive and difficult.
4. Seems to be able to handle greater volumes of data than Xharvester;
5. Not user friendly.
6. Can improve the style of presentation. 1- Due to the amount of information presented the protocol documentation could benefit from including more diagrams. 2- Similarities/differences between other formats would be better presented in a table to help the user follow.

7. While it seems to be more difficult to implement, the wider range of functions are an advantage to the protocol.

8. Too much information, not conveyed in a way that is easy to understand, however towards the ends of the documentation good descriptions of how everything works were given. Some portions (request) were difficult to understand, specifically everything to do with sets.

9. Needs to cater for selective output files, allows users to select what they want the output to look like, as opposed to the current complicated XML file.

10. Allow the option to locate a single record, and not just the set it is in (From what I read, it did not look like you could locate a record by way of its identifier).

Some of the comments suggest that while participants seem to have appreciated a more simplistic approach (see Xharvester comment list, items1, 5, 6, 8) to protocol design, they are not willing to compromise on functionality (see Xharvester comment list items 2, 4, 7). It seems that there is such a thing as too much simplicity.
In other comments is is obvious that either participants did not read the whole documentation, or because this was the first time they learnt about protocols they did not really understand them well. This is based on the fact that some of the claims made were not correct, for example item 10 on the OAI-PMH comments list.
In the case of comments such as 9 on the OAI-PMH comment list, they can be the result of a lack of knowledge about the area and the technologies used, or because protocol designers assume that there are things everyone trying to implement a protocol should know, and in doing so leave out information that may be important to new users, such as the fact that the use of XML is appropriate for inter-systems communications.
Based on the comments, this part of the evaluation concludes that the best approach to protocol design is to change the way things are currently done by introducing simplicity in the design guidelines. Simplicity should not be something left for after the design is complete, but rather it is a criteria that should be followed in every aspect of the design and process, thus making implementation and adherence easier.

Xharvester vs. OAI-PMH Conclusion
The overall conclusion from the comparison of Xharvester and OAI-PMH is similar to the Xsearch vs. SRU conclusions. Xharvester also proves to be simpler to understand as well as more usable in terms of writing and presentation style than OAI-PMH. The ratings of the perceived degree of implementation difficulty between the two protocols indicate that, according to the participants, neither one of the implementations would be an easy task. Such result is surprising in the case of Xharvester due to the fact that overall it was rated simple to understand and easy to follow. As in the case of SRU, OAI-PHM can also benefit from some simplification on how the data/information is presented to possible users.

5.2.1.3 HIP-CF Complete Evaluation

HIP-CF complete evaluation was conducted the same way as the different Xscarch vs. SRU and Xharvester vs. OAI-PMH evaluations. The difference in this case is that the participants were given double the time and more work.
This part of the evaluation involved 4 Masters students from two different research laboratories. Each student was given the documentation of 3 protocols, namely: OAI-PMH, SRU and HIP-CF (the complete HIP-CF documentation contains the preamble and documentation of 3 different services, Xbrowse, Xharvester and Xsearch), also a consent form (see appendix D) and a questionnaire (see appendix E2).
As in the two previous sections, Xharvester was compared to OAI-PMH and Xsearch was compared to SRU. Unfortunately Xbrowse could not be compared to another high-level interoperability protocol for browsing because there is no browsing protocol formal documentation available.
The purpose of this evaluation was to analyse the protocol as a whole and the idea of a suite of protocol services as opposed to individual protocols. The results of this evaluation are presented below.

Programming Skills: the 4 participants described their programming skills as follows:

- 1 participant said that he/she is an expert programmer.
- 1 participant said that he/she is a good programmer.
- 2 of participants said that they are average programmers.

Previous knowledge of high-level interoperability protocols: When asked if they knew any high-level interoperability protocols prior to this evaluation, 3 participants responded yes and 1 participant responded no.

Findings: 75% of participants said that they had prior knowledge with high-level interoperability protocols. The protocols mentioned were HTTP by one participant and OAI-PMH by the other two participants.

Level of Understanding Ratings: The participants were asked to compare their level of understanding of the three protocols, on a scale of 1 to 10 where 1 is 'I still don't understand the protocol' and 10 is 'I now have a good understanding of the protocol'. Figure 5.10 shows the results for the levels of understanding ratings of HIP-CF, OAI-PMH and SRU.
Findings: HIP-CF - All participants rated their levels of understanding of HIP-CF above the average level.
With 100% above average, the results indicate that the users' understanding HIP-CF ranges from relatively easy to really easy.
Findings: OAI-PMH - was rated at average or above average by all participants.
Note that the two participants that rated OAI-PMH at level 10 are the two whose previous knowledge of interoperability protocols was exactly in working with OAI-PMH. The two participants who rated their understanding at average and just above average, levels 5

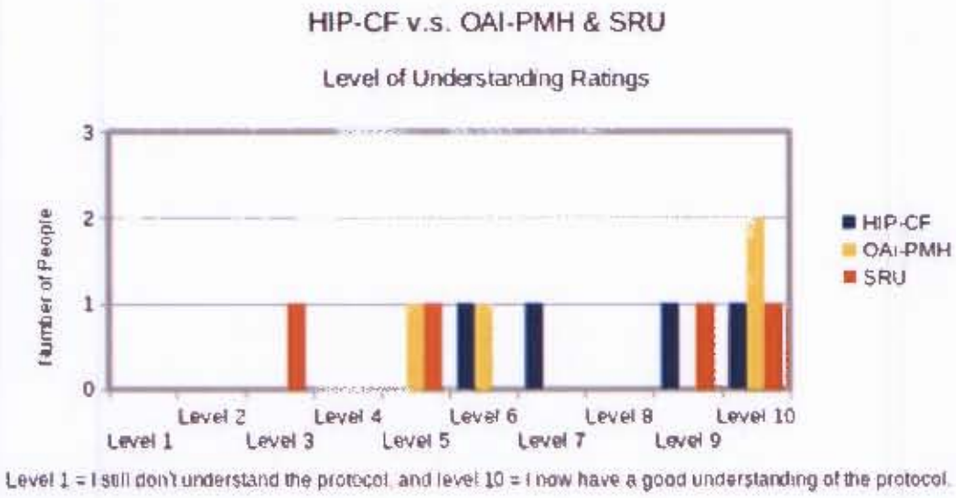

Figure 5.10: Protocol Suite vs. Individual Protocols Levels of Understanding Ratings

and 6, were not familiar with OAI-PMH prior to the evaluation. With 75% above average, OAI-PMH's understanding also ranges from relatively easy to really easy.
Findings: SRU - was the only protocol to get a below average rating. The ratings for this protocol were divided by about 50% between really easy to understand and average or below, suggesting that it is easier for people with knowledge of high-level interoperability protocols to understand it, than it is for those with no prior knowledge.

Writing and Presentation Style Ratings: The participants were asked to compare the writing and presentation styles of the three protocols, on a scale of 1 to 10 where 1 is 'poor' and 10 is 'very good'. Figure 5.11 shows the results for the writing and presentation style ratings of HIP-CF, OAI-PMH and SRU.
Findings: HIP-CF - 1 participant rated this category below average and the other 3 rated it above average.
With 75% above average the results indicate that HIP-CF's writing and presentation style is well accepted by the participants, suggesting that it is simple and easy to follow and comprehend.
Findings: OAI-PMH - 1 participant rated this category average and the other 3 all rated it above average.
None of the participants rated this category below average. This result indicates that OAI-PMH's writing and presentation style is also well-accepted, making it simple and easy to follow and comprehend.
Findings: SRU - 1 participant rated this category below average, 1 participant rated it average and the other 2 participants rated it above average.
SRU ratings for writing and presentation style were balanced between average and above average, with 50% of participants rating it as average and below and the other 50% rating

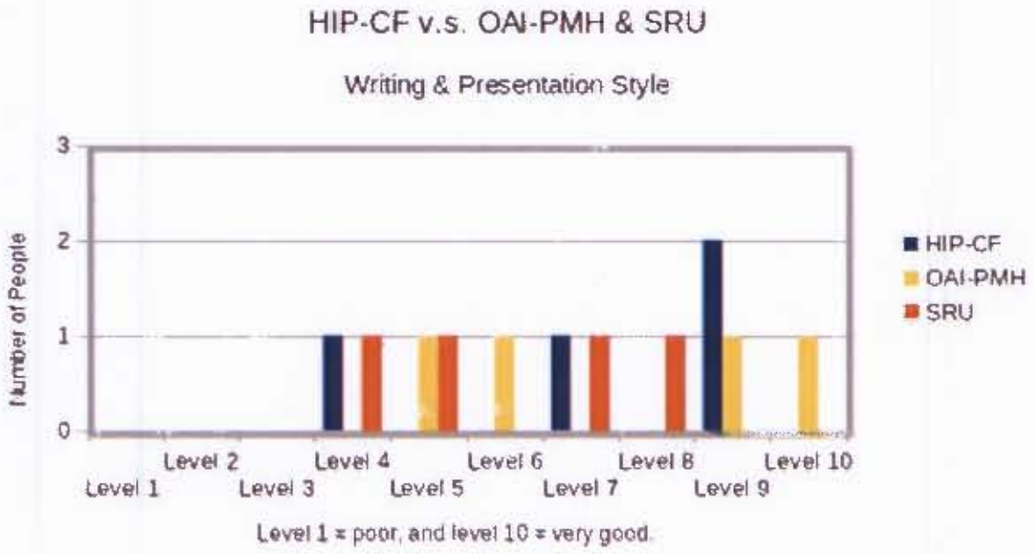

Figure 5.11: Protocol Suite vs. Individual Protocols Writing and Presentation Style

it above average. While its lowest rating is the same as the HIP-CF lowest rating, its highest rating is not as high as the other protocols highest ratings.
Participants rated what they know (OAI-PMH) or HIP-CF higher than the other option.

Perceived Degree of Implementation Difficulty: The participants were asked to rate how difficult they think the implementation of the protocols would be, on a scale of 1 to 10 where 1 is 'easy' and 10 is 'difficult'. Figure 5.12 shows the results of the perceived degree of difficulty associated with the implementations of HIP-CF, OAI-PMH and SRU.

Findings: HIP-CF - 1 participant rated the perceived degree of difficulty associated with implementing HIP-CF as an average task and the other 3 participants rated it above average.
With 75% of ratings above average, 2/3 of it at level 9, HIP-CF's implementation is perceived as difficult.
Findings: OAI-PMH - 1 participant rated the perceived degree of difficulty associated with implementing OAI-PMH below average and the other 3 rated it above average.
The results indicate that as with HIP-CF, OAI-PMH implementation was considered above average difficult by most participants, in this case also by 75% of participants, although at different levels.
Findings: SRU - 1 participant rated the perceived degree of difficulty associated with implementing SRU below average and the other 3 participants rated it above average difficult.
Similar to the other two protocols, SRU results also indicate that 75% of participants perceive implementation to be an above-average difficult task.
All protocols are perceived to be difficult to implement.

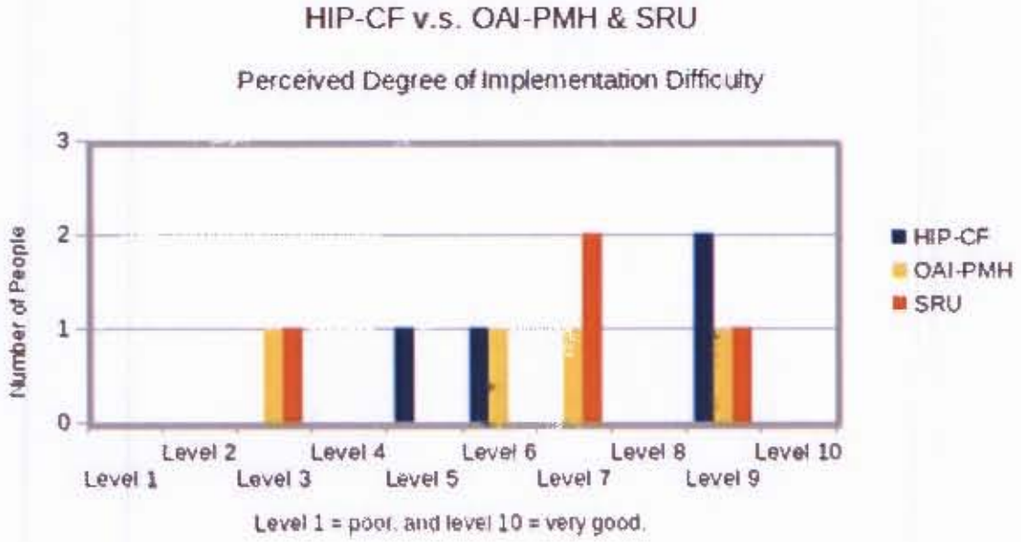

Figure 5.12: Protocol Suite vs. Individual Protocols Degree of implementation Difficulty Ratings

Correlation between participants programming skills and their responses to the questionnaire, and their opinions on a suite vs. individual protocols: Table 5.3 show the participants' self described level of programming skills and their choices for each each main comparison category.

Programming Level	Q 6 P1	Q 6 P2	Q 6 P3	Q 7 P1	7 P2	Q 7 P3	Q 8 P1	Q 8 P2	Q 8 P3	Q 10
Expert	7	6	5	9	5	4	9	7	7	A
Good	10	10	10	7	9	8	5	3	7	A
Average 1	6	5	3	4	6	5	6	6	3	B
Average 2	9	10	9	9	10	7	9	9	9	A

Table 5.3: Participants' programming skills and their answers. P1 = HIP-CF, P2 = OAI-PMH and P3 = SRU

Correlation:

- Question 6 - Understandability:
  - The *expert programmer* rated HIP-CF at level 7, OAI-PMH at level 6 and SRU at level 5. This participant had no previous knowledge of high-level interoperability protocols. His/her understanding of each of the protocols has at most a 2 level difference, with HIP-CF being the one he/she understood more of and SRU being the one that was least understood.
  - The *good programmer* rated his/her understanding of all protocols at level 10, meaning that he/she understood all of them very well. Note that this participant had previous knowledge of high-level interoperability protocols and

had already implemented OAI-PMH. This may be the contributing factor for his/her ability to understand the other protocols. .

- The *average programmers* both had previous knowledge of high-level interoperability protocols - they rated their understandability very differently with average 1 rating between levels 3 and 6, specifically HIP-CF 6, OAI-PMH 5 and SRU 3, and average 2 rating HIP-CF 9, OAI-PMH 10 and SRU 9.
  * Programming skills did not seem no have much of an impact here - what did is the previous knowledge for the two participants who had worked with OAI-PMH before. They (good and average 2) best understood all the protocols. Although the expert programmer did not have previous knowledge of high-level interoperability protocols, he/she understood the protocols better than the average programmer with previous knowledge of high-level interoperability protocols (average 1). So in the case of the participants who had previously implemented a high-level interoperability protocol, previous knowledge had a stronger impact than programming knowledge and it can be speculated that for the participants who never implemented a high-level interoperability protocol programming level had a bigger impact than previous knowledge.

- Question 7 - Writing and presentation style:
  - The *expert programmer* rated the writing and presentation style of the three protocols at the following levels: HIP-CF 9, OAI-PMH 5 and SRU 4. Although at different levels, this is the same decreasing order given for understandability, with HIP-CF rated the highest, followed by OAI-PMH and SRU the lowest. This time the difference between the levels is higher, of at least 4 levels between protocols.
  - The *good programmer* rated the writing and presentation style of the three protocols at the following levels: HIP-CF 7, OAI-PMH 9 and SRU 8. The difference between the levels was not as big: OAI-PMH was rated the highest, followed by SRU and HIP-CF was rated the lowest.
  - The *average programmers* rated the writing and presentation style of the three protocols very differently. average 1 rated HIP-CF 4, OAI-PMH 6 and SRU 5, while average 2 rated HIP-CF 9, OAI-PMH 10 and SRU 7. They had the same level of programming skills and previous knowledge of interoperability protocols, but once again average 2's previous work with OAI-PMH seems to put him/her ahead with the highest ratings for all protocols.
    * The overall best ratings for writing and presentation style was given to OAI-PMH, followed closely by HIP-CF and SRU got the lowest ratings. Again, previous knowledge is a big factor on participants' ratings and level of programming skills does not have a very big impact.

- Question 8 - Implementation difficulty:
  - The *expert programmer* rated the perceived degree of implementation difficulty associated with the three protocols at the following levels: HIP-CF 9, OAI-PMH 7 and SRU 7. The expert programmer understood HIP-CF better than

the other protocols, and rated its writing and presentation style simpler but rated it as the hardest protocol to implement, at level 9. He/she perceives the implementations of OAI-PMH and SRU to be equally difficult, at level 7, but still easier than that of HIP-CF.

- The *good programmer* rated the perceived degree of implementation difficulty associated with the three protocols at the following levels: HIP-CF 5, OAI-PMH 3 and SRU 7. Not surprisingly, as this participant has already implemented OAI-PMH, he/she finds its implementation to be an easy task, at level 3. For the good programmer, the difficulty of implementing HIP-CF is average and SRU is almost half-way between average and difficult.
- Once again the *average programmers* rated the perceived degree of implementation difficulty associated with the three protocols quite differently. Average 1 rated HIP-CF 6, OAI-PMH 6 and SRU 3, and average 2 rated all three protocols at level 9. Although he/she has already implemented OAI-PMH, average 2 still finds it difficult, and he/she believes that the implementation of any of the protocols would be equally difficult. Average 1 does not find the implementation as difficult as average 2. In fact, for average 1 the implementation of SRU would be easy, and the implementation of either HIP-CF or OAI-PMH would be just above average difficult.
    * Assuming that previous knowledge of the subject has a bigger impact than programming skills when it comes to implementing protocols would be acceptable at this point given the results from users. Although an expert is expected to find programming exercises easier than a good programmer and a good programmer should find them easier than an average programmer, this evaluation has showed the opposite for most cases. This can be a result of poor self-qualification of the participants - programming skills or, as said above, previous knowledge does have a stronger impact than programming skills when it comes to rating the expected level of difficulty in solving a problem.
      An interesting fact to note is that even one of the participants who has previously implemented OAI-PMH (average 2) rated it as above average difficult. It may be confusing to understand the reason behind this since this participant had ratings which suggested that all protocols were easy for him/her to understand and that he/she was satisfied with the writing and presentation style. An assumption is that average 2 may have considered that greater understandability and flexibility at the surface level may have concealed higher complexity at the implementation level.

A suite of protocols vs. individual protocols: The participants were asked to give their opinion on whether is it better to have individual protocols (which is the current situation) or have a suite of protocols, i.e. one protocol that supports multiple high-level interoperability services (proposed in this research).As seen on table 5.3 question 10 (see Appendix E2), 3 participants chose option A and 1 participant chose option B. The results indicate that all participants agree that a suite of protocols is a better than having individual protocols.

A suite of protocols would be a great idea because when a programmer implements one service he/she would have to read the protocol (preamble and service documentation), so if the same programmer wanted to implement a difference service from the same suite, previous knowledge of the protocol would be an advantage.
General comments: Participants were asked to make comments about the protocols. See the comments followed by a brief discussion below.
HIP-CF

1. Easier to read than the other two.
2. Documentation is abstract and not very detailed.
3. One request-response may make implementation more complex than multiple request-response pairs.
4. Multi-service support is an advantage.
5. Not very concise, hence certain implementations to the developer which may involve inconsistencies.
6. Xbrowse sounds very tightly coupled.
7. Would benefit from having an XML style sheet for validating the XML.
8. The diagrams, tables, images, and terminology make it easy to understand.

OAI-PMH

1. Not hard to read.
2. Very detailed explanations, at low levels.
3. Not easy to fully understand but implementation is made easier by the level of discussion.
4. Simplistic.
5. The standards used are well explained.
6. Very limited services.
7. Overwhelming amount of text makes it hard to read.
8. Most examples were easy to understand, however it is unnecessary for a single example to span over multiple pages.
9. Newly introduced concepts were highlighted.

SRU

1. Difficult to understand, but the protocol model makes it easier to understand.
2. Requires previous knowledge of the work in order to follow the explanation.

3. Good explanation of how the URL parameters are constructed.
4. Lacks conciseness in certain implementation areas.
5. The language used in a bit inconsistent.
6. The query language can be very complicated for simple but multi-parameter searches.
7. SOAP support is an advantage.
8. Limited services.
9. Rigid Components.
10. Good examples, good one-liner explanations for the different sections.
11. Could benefit from adding images and diagrams.

The comments indicate that there were mixed feelings towards the protocol presentation styles, with participants saying that HIP-CF is easy to read and the tables, images and diagrams are an advantage but also saying that it is abstract and the lack of conciseness may lead to inconsistencies. One thing that the participants agreed on was the fact that a multiple services approach is an advantage. One user suggests an XML validator for HIP-CF but that would not be possible as the XML response may contain or not contain any values chosen by the implementers, and any kind of validator would create unnecessary and unwanted constraints.
There were comments about OAI-PMH being too long and hard to read with unnecessarily long examples, and there were comments saying that it is not hard to read it, that OAI-PMH is simplistic and that the explanations are very detailed and that the level of discussion made it easier to understand. Again mixed reviews. SRU was said to be difficult to understand, require prior knowledge of the subject, rigid and also that it is not concise, but participants appreciated the protocol model, the fact that it is SOAP based, and the quality/type of examples.
According to these results a lot can still be changed to improve the protocol documentation.

HIP-CF Complete Evaluation Findings
The results suggest that OAI-PMH is the simplest of the three protocols, followed by HIP-CF and then SRU.
It is also noted that previous high-level interoperability protocol knowledge is an important factor for participants/possible users to understand any protocol. This is evident from the two participants who were familiar with OAI-PMH - this clearly helped them in understanding the other protocols, that was noticeable by their scores to all protocols which were in most cases higher/better than those of the other two participants.
Also important id the fact that all participants agree that a protocol suite approach is good. So it can be said that ideally high-level interoperability protocol services should be part of a suite of protocols and at least as simple as OAI-PMH.

Overall Understandability Evaluation Conclusion
The participants who had never worked/used any of the protocols evaluated here found HIP-CF easier to understand when compared to the other protocols, for both single services evaluations, and the participants who had experience in working with any of the protocols evaluated here found the protocol they are familiar with easier to understand, specifically OAI-PMH simpler than HIP-CF and SRU.
The idea of a protocol suite was well received and approved by participants.
The participant ratings on all questions were affected more by their previous knowledge of the area than by their self-reported level of programming skills.

It may be time to change the way things have been done until now and merge all the efforts towards facilitating high-level interoperability to form a "super protocol" that supports all services. And in doing so, simplicity should not be accommodated during the "How To Improve This" phase, but should rather be one of the main design requirements.

5.3 Entropy

The field of information theory introduces the concept of Entropy (E). Entropy is a measure of the amount of order or predictability in a message [8]. The value of entropy is a small number when there is a lot of order and it is a large number when there is a lot of disorder. Entropy is directly related to data compression, in the sense that ideally the length of a message after it is encoded should be equal to its entropy. This is a measure of the quantity of information or the information content.

Why measure entropy?
The value of entropy is equal to the minimum number of bits necessary to encode a message without losing any valuable information. This number helps eliminate non-crucial information from the message, such as pieces of information that have a probability of 1 because they do not change. They are always present as long as the message format remains the same, for example, the <html> and <body> tags in an .html file. Elements with a probability of 1 have an entropy of 0.
Using the entropy value, a message can be compressed to obtain the representation of the data file that occupies less space but preserves all the information [8].

5.3.1 Entropy Calculations

The overall entropy is the average of the entropy of the individual probabilities occurring. It is calculated by the following formula:

$E = -\sum_i^n P_i \ logP_i \ bits.$ [8]

Where:

- E is entropy;
- the sign (-) is used to obtain a positive value, due to the fact that if a probability is less than 1 it will always have negative logarithms;

- $\sum_i^n$ = the sum of all probabilities from i to n;
- P = the probability of an event occurring, such that P_i = the probability of event *1* occurring and P_n = the probability of event n occurring;
- *log* = the exponent a fixed/base number is raised to, in order to produce a given value; in this case the exponent is 2;
- *bits* is the unit of measurement.

In other to calculate the entropy of messages obtained from the different protocols, some restrictions will be made. This is to create a common base line for all protocols, and also because the size of the message, particularly big messages, can lead to cumbersome calculations.

5.3.1.1 Facts and Assumptions

All individual items in the response are used in the calculation. The tags will be calculated as a whole but the elements inside the tags will be calculated as individual characters.

Tags

Tags are usually divided into two categories: mandatory tags and optional tags. A mandatory tag will always occur and therefore has a probability of 1, while an optional tag may or may not occur and therefore has a different probability. A tag may occur more than once if it is repeatable. Both mandatory an optional tags can be repeatable with no limitations as to the number of times a specific tag may be repeated.

Elements within the tags

Consider a gaming application that creates XML files to store the names of players. Before playing the game the player must write their name which is then stored as the following XML file : <name>Jorgina83</name>. In this file the tag "<name>" will always occur and it will never change, therefore it has a probability of 1. The same is true for "</name>". However the name within the tags will change, so the probability of it being "Jorgina83" is not 1. for elements that change the entropy calculation involves each individual character of that element. In the case of the name Jorgina83, one would ask what is the probability of the first character being a J? There are 26 letters in the English alphabet, each of these letters can be used in upper or lower case, creating a total of 52 possible letters, and there are ten possible numbers from 0-9.

There is a very large number of possible punctuation marks, typography, and currency & intellectual property symbols, but to keep the calculations manageable, our allowed character set will only include the following punctuation characters: space [ ], comma [,], full stop [.], hyphen [-], two hyphenminuses [–], quotation marks [" and "], colon [:], semi-colon [;], slash [/], brackets [(and)], percentage [%], underscore [_] and terminating character [Ø][2].

It is acceptable for a character set to be restricted to a specified number of characters. In his "Prediction and Entropy of Printed English"[77] paper, C. E. Shannon only used the 26 letters of the alphabet; Hamid Moradi et al.6 [53] used the 26 letters of the alphabet

[2]The characters chosen were the ones that appeared in either one or both XML files used in the search entropy calculations plus the terminating character which is necessary to know where a string ends which is important to know when decoding the encoded message.

and a space character; and .William J. Paisley's character set included the 26 letters of the alphabet plus the space character and the full stop punctuation mark [64] in his entropy calculations.
The allowed character set for this research will be comprised of a total of 77 possible characters. Therefore the probability of the first character being a J is 1/77. That probability is calculated for each individual character for elements inside a tag that are known to change.
Th calculation are done keeping within certain simplistic assumptions, such as the fact that each character within a tag will have the same probability. In a real life situation this would not be the case, as the encoding of each individual character will have a different probability of occurring at a certain place given its use (punctuation characters) and given the previous character(s) certain letters and/or numbers would have a higher or lower probability than others. Another issue that would have to be considered in a real life scenario is the case of hierarchical entropy, where the calculation would involve elements presented in a hierarchical manner and nested within others.

5.3.1.1.1 Search Entropy: Similar requests are sent using Xsearch and SRU - Similar not in format or how they are processed but similar in the fact that in both cases a request is sent to retrieve a record that matches the query term "interoperability" and the number of matching records in the response was restricted to 1. Figures 5.13 and 5.14 show the XML response files obtained from each of the requests:

```
<response>
<total_matches>2</total_matches>
<record>
<title>Achieving interoperability in critical IT and communications systems /</title>
<creator>Desourdis, Robert I.</creator>
<description>Includes bibliographical references and index</description>
<identifier>06482XYZ</identifier>
<keyword>Interoperability</keyword>
</record>
</response>
```

Figure 5.13: Xsearch response file used to calculate entropy

Figure 5.14: SRU response file used to calculate entropy

Xsearch

The XML file returned from an Xsearch request, and the probabilities associated with the different elements are shown below:

<response> Mandatory tag (M), probability of the string or tag (P) = 1

<total_matches> M, P = 1.

2∅ P = 1/10. The total number of matching records will always be a numeric value, and since there are only 10 possible options for numeric values the probability of that number being a 2 is 1/10. in cases where there is more than one digit the calculation is done for each individual digit. ∅ has a probability of 1/77.

</total_matches> M, P = 1.

<record>M, P = 1/2 * 6. There is a 50/50 chance of a matching record being available.

<title> Optional (O), P = 1/15 * 5. Title is one of the 15 optional and repeatable DC elements.

Achieving interoperability in critical IT and communication systems /∅ P = 1/77 * 69. The probability of each individual character is 1/77 because there are 77 characters in the allowed character set, and that number is multiplied by 50 because there are 50 characters in the title. ∅ has a probability of 1/77.

</title> O, P = 1/15 * 6.

<creator> O, P = 1/15 * 7.

Desourdis, Robert I.∅ P = 1/77 * 20. ∅ has a probability of 1/77.

</creator> O, P = 1/15 * 8.

<description> Optional, P = 1/15 * 11.

Includes bibliographical references and index Ø P = 1/77 * 46. Ø has a probability of 1/77.
</description>O, P = 1/15 * 12.
<identifier>O, P = 1/15 * 10.
06482XYZØ P = 1/77 * 8. Ø has a probability of 1/77.
</identifier>O, P = 1/15 * 11.
<keyword>M, P = 1.
InteroperabilityØ P = 1/77 * 16. Ø has a probability of 1/77.
</keyword>M, P = 1.
</record>M, P = 1/2 * 7.
</response>M, P = 1.

Using the formula: $E = \quad \sum_i^n P_i \ logP_i \ bits$

$= (\log_2(\frac{1}{2}) * (\quad (6 + 7))) + (log_2(\frac{1}{10}) * (-1)) + (log_2(\frac{1}{15}) * (-(5 + 6 + 7 - 8 + 11 + 12 - 10 - 11))) + (log_2(\frac{1}{77}) * (-(1 + 69 - 1 + 20 + 1 + 46 + 1 + 8 + 1 - 16 + 1)))$

$= (\log_2 2 * (6 + 7)) - (log_2 10 * (1)) + (log_2 15 * (5 + 6 + 7 + 8 + 11 - 12 + 10 - 11)) \cdots (log_2 77 * (1 + 69 + 1 + 20 + 1 + 46 + 1 + 8 - 1 - 16 - 1))$

$= (\log_2 2 * 13) - (log_2 10 * 1) + (log_2 15 * 70) - (log_2 77 * 165)$

$= 13 - 3.3219280948874 \cdots 273.482341692595 + 1034.0197792146585$

Entropy = 1324 bits.

According to Shannon[77], the entropy of encoding one character is the log of the total number of probability options, in this case for characters with a probability of 1/77 the log of 77 = 6 bits per character for each character in the allowed set; for characters with a probability of 1/15 the log of 15 = 4 bits per character for each optional tag, log of 10 or 3 bits for each numeric character, and log of 2 = 1 bit per character for characters in tags with a 50/50 chance of occurrence. The mandatory tags have a probability of 1 and therefore an entropy of 0, so they are not encoded. Shannon's theory can be confirmed from the numbers obtained above.
The total entropy for characters with a probability of 1/77 is 1034.0197792146585, this number divided by 165, which is the total number of characters with a probability of 1/77 = 6.26678654O6949, which approximates to 6 bits per character.
The total entropy for characters with a probability of 1/15 is 273.482341692595, this number divided by 70, which is the total number of characters with a probability of 1/15 = 3.9068905956085, which approximates to 4 bits per character.
The total entropy for characters with a probability of 1/10 is 6.6438561897748, this number divided by 2, which is the total number of characters with a probability of 1/10 3.3219280948874, which approximates to 3 bits per character.
The total entropy for characters with a probability of 1/2 is 13, this number divided by 13, which is the total number of characters with a probability of 1/2 = 1, which approximates to 1 bit per character.

SRU

The XML file returned from an SRU request, and the probabilities associated with the different elements are shown below:

<zs:searchRetrieveResponse> M, P = 1.

<zs:version> M, P = 1.

1.1Ø P = 1/77 * 3. Ø has a probability of 1/77.

</zs:version> M, P = 1.

<zs:numberOfRecords> M, P = 1.

235Ø P = 1/10 * 3. Ø has a probability of 1/77

</zs:numberOfRecords> M, P = 1.

<zs:records> M, P = 1.

<zs:record> M, P = 1/2 * 9. There is a 50/50 chance of a matching record being available.

<zs:recordSchema> M, P = 1.

info:srw/schema/1/dc-v1.1Ø P = 1/77 * 25. Ø has a probability of 1/77.

</zs:recordSchema> P = 1.

<zs:recordPacking> P = 1.

xmlØ P = 1/77 * 3. Ø has a probability of 1/77.

</zs:recordPacking> P = 1.

<zs:recordData> P = 1.

<srw_dc:dc xsi:schemaLocation="info:srw/schema/1/dc-schema http://www.loc.gov/standards/sru/resources/dc-schema.xsd"Ø> P = 1/77 * 115. Ø has a probability of 1/77. The probability of this tag is calculated differently from other because the content within changes with the metadata format.

<title> P = 1/15 * 5.

Achieving interoperability in critical IT and communication systems /Ø P = 1/77 * 69. Ø has a probability of 1/77.

</title> P = 1/15 * 6

<creator> P = 1/15 * 7.

Desourdis, Robert I.Ø P = 1/77 * 20. Ø has a probability of 1/77.

</creator> P = 1/15 * 8.

<type> P = 1/15 * 4.

textØ P = 1/77 * 4. Ø has a probability of 1/77.

</type> P = 1/15 * 5.

<publisher> P = 1/15 * 9.

Boston : Artech House,Ø P = 1/77 * 22. Ø has a probability of 1/77.

</publisher> P = 1/15 * 10.

<date> P = 1/15 * 4.

2009.Ø P = 1/77 * 5. Ø has a probability of 1/77.

</date> P = 1/15 * 5.

<language> P = 1/15 * 8.

engØ P = 1/77 * 3. Ø has a probability of 1/77

</language> P = 1/15 * 9.

<description> P = 1/15 * 11.

Includes bibliographical references and index.Ø P = 1/77 * 46. Ø has a probability of 1/77

</description>P = 1/15 * 12.
<subject>P = 1/15 * 7.
Internetworking (Telecommunication)∅ P = 1/77 * 35. ∅ has a probability of 1/77.
</subject>P = 1/15 * 8.
<subject>P = 1/15 * 7.
Emergency management Communication systems–Computer networks.∅ P = 1/77 * 63. ∅ has a probability of 1/77.
</subject>P = 1/15 * 8.
<identifier>P = 1/15 * 10.
URN:ISBN:9781596933897∅ P = 1/77 * 22. ∅ has a probability of 1/77.
</identifier>P = 1/15 * 11.
<identifier>P = 1/15 * 10.
URN:ISBN:1596933895∅ P = 1/77 * 19. ∅ has a probability of 1/77.
</identifier>P = 1/15 * 11.
</srw_dc:dc∅>P = 1/77 * 10. ∅ has a probability of 1/77.
</zs:recordData>P = 1.
<zs:recordPosition>P = 1.
1∅ P = 1/10. ∅ has a probability of 1/77.
</zs:recordPosition>P = 1.
</zs:record>M, P = 1/2 * 10.
</zs:records>P = 1.
</zs:searchRetrieveResponse>P = 1.

Using the formula $E = -\sum_i^n P_i \ logP_i \ bits$

$= (\log_2(\frac{1}{2})*(-(9-10)))+(log_2(\frac{1}{10})*(-(3+1)))-(log_2(\frac{1}{15})*(-(5+6+7+8-4+5+9-10-4-5\cdot 8\dot{+}9+11+12+7+8+7\dot{+}8\dot{+}10\dot{+}11\cdot 10-11)))\cdot (log_2(\frac{1}{77})*(-(3+1+1+25+1+3-1+115+1\dot{+}69\dot{+}1\dot{+}20\dot{+}1+4-1+22+1+5-1+3-1-46+1+35+1-63-1+22+1+19-1+10+1+1)))$

$= (\log_2 2*(9+10))\cdot (log_2 10*(3+1))+(log_2 15*(5-6\dot{+}7\dot{+}8\div 4+5+9\cdot 10+4\div 5+8-9+11\dot{+}12\dot{+}7-8+7+9+10+11-10+11))-(log_2 77*(3+1+1-25+1-3+1+115-1+69+1+20-1+4-1+22+1-5+1+3-1+46+1-35+1-63+1+22+1+19+1\dot{+}10-1+1))$

$= (\log_2 2 * 19) + (log_2 10 * 4) + (log_2 15 * 175) - (log_2 77 * 482)$

$= 19 + 13.2877123795496 \div 683.7058542314875 + 3020.5911126149418$

Entropy = 3737 bits.

The entropy of any two different files will never be the same. It will depend on the size of the file and on the amount of "unpredictable" information, which is the information that does not have a probability of 0. The Xsearch file contains less information and therefore its entropy value is lower than that of SRU. However both files can be encoded in the same manner.

5.3.1.1.2 Harvesting Entropy Calculations: The calculations were done on two XML files obtained as responses to the harvesting *ListRecords* request.Figures 5.15 and 5.16 show the XML response files obtained from each of the requests.

Xharvester
Below is the entropy calculation for the XML file returned from an Xharvester request (see Figure 5.15).

Using the formula $E = -\sum_i^n P_i \ logP_i \ bits$

$= (\log_2(\frac{1}{2}) * (\ \ (6 \mid 7))) \mid (log_2(\frac{1}{10}) * (\ \ 11)) \cdot (log_2(\frac{1}{15}) * (-(5+6-6+7-6+7+6+7+6-7-6-7-11-12-10+11-4-5+4+5))) - (log_2(\frac{1}{77}) * (-(11-1-19 \mid 1+2+1 \mid 115 \mid 1 \cdot 16 \mid 1 \cdot 17 \mid 1-20+1-15-1-14-1-1162+1+8+1+19+1+10-1+1)))$

$= (\log_2 2 * (6-7)) + (log_2 10 * (11)) - (log_2 15 * (5+6+6-7+6 \cdot 7 \mid 6 \mid 7-6 \mid 7 \mid 6 \mid 7+11 \cdot 12 \mid 10 \cdot 11-4+5+4-5)) + (log_2 77 * (11+1+19+1+2-1+115+1+16+1+17+1+20+1+15 \cdot 1 \mid 14 \cdot 1 \mid 1162 \cdot 1 \mid 8+1+19 \mid 1-10-1+1))$

$= (\log_2 2 * 13) \mid (log_2 10 * 11) + (log_2 15 * 138) - (log_2 77 * 1442)$

$= 13 + 36.5412090437614 - 539.150902193973 - 9036.7061916820458$

Entropy = 9625 bits.

OAI-PMH
Below is the entropy calculation for the XML file returned from an OAI-PMH request (see Figure 5.16).

Using the formula $E = -\sum_i^n P_i \ logP_i \ bits$

$(\log_2(\frac{1}{2}) * (-(6+7))) - (log_2(\frac{1}{15}) * (-(8-9+10+11+10+11 \mid 10 \mid 11 \cdot 10 \mid 11+10+11+10-11+10-11+10+11-14+15-7+8+7+8+13+14))) + (log_2(\frac{1}{77}) * (\ \ (113 \mid 1+20 \mid 1 \mid 69 \cdot 1 \mid 34+1+11 \mid 1 \mid 32+1+10+1+18-1+20+1+25+25+1+25-1+26-1+121-1+114-1 \mid 16 \mid 1 \mid 17+1 \mid 20+1+15 \mid 1 \mid 14 \mid 1+29+1+37-1+27 \cdot 1-1162 \cdot 1-10+1+19+1+42+1+10+1+19+1)))$

$= (\log_2 2 * (6-7)) + (log_2 15 * (8+9+10-11-10+11+10+11-10-11-10-11+10 \mid 11 \mid 10 \cdot 11+10+11+14 \mid 15 \cdot 7+8 \mid 7 \mid 8-13 \mid 14)) \cdot (log_2 77 * (113+1-20+1+69+1-34-1-11+1+32+1+10+1+18+1+20+1 \cdot 25-25-1+25 \mid 1 \cdot 26 \mid 1 \cdot 121+1+114 \cdot 1+16-1+17+1+20-1+15+1+14-1+29-1+37-1+27+1-1162+1-10+1+19+1 \cdot 42 \mid 1 \mid 10 \mid 1 \cdot 19 \cdot 1))$

$= (\log_2 2 * 13) - (log_2 15 * 271) + (log_2 77 * 2128)$

$= 13 \mid 1058.7673514099035 \mid 13335.7217585987472$
Entropy = 14407 bits.

Like in the search entropy calculation the value will always be higher for the file with the highest information content. In all cases it would be possible to encode and decode the files without the loss of data. For the harvesting calculations the Entropy of OAI-PMH is almost 50% higher than the Entropy of Xharvester.
For both search and harvesting entropy calculations the HIP-CF services have shown considerably lower number of bits necessary to encode its' messages, than it is necessary to encode the messages of both SRU and OAI-PMH. Thus indicating that HIP-CF can provide a better data compression ratio.

<HIP-CFxharvester>
<request>ListRecords</request>
<responseDate>2012-Feb-11 2:43:36</responseDate>
<metadataFormat>DC</metadataFormat>
<record>
<title>Conformational properties of two exopolysaccharides produced by Inquilinus limosus, a cystic fibrosis lung pathogen</title>
<author>Kuttel, Michelle</author>
<author>Ravenscroft, Neil</author>
<author>Foschiani, Michaela</author>
<author>Cescutti, Paola</author>
<author>Rizzo, Roberto</author>
<description>
Inquilinus limosus is a multi-resistant bacterium found in the respiratory tract of patients with cystic fibrosis. This bacterium produces two unique fully pyruvylated exopolysaccharides in similar quantities: an alpha-(1->2)-linked mannan and a beta-(1->3)-linked glucan. We employed molecular modelling methods to probe the characteristic conformations and dynamics of these polysaccharides, with corroboration from potentiometric titrations and circular dichroism experiments. Our calculations reveal different structural motifs for the mannan and glucan polysaccharides: the glucan forms primarily right-handed helices with a wide range of extensions, while the mannan forms only left-handed helices. This finding is supported by our circular dichroism experiments. Our calculations also show that the (1->3)-beta-D-Glcp linkage is more dynamically flexible than the (1->2)-alpha-D-Manp; the glucan characteristically forms a range of wide helices with large central cavities. In contrast, the mannan forms rigid regular 'bottlebrush' helices with a minimal central cavity. The widely different character of these two polymers suggests a possible differentiation of biological roles.
</description>
<identifier>16576XYZ</identifier>
<type>Journal (Paginated)</type>
<date>2012-03-01</date>
</record>
<resumptionToken>52890760129</resumptionToken>
</HIP-CFxharvester>

Figure 5.15: HIP-CF ListRecords Response

```
<OAI-PMH xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/ http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
<responseDate>2012-02-12T19:56:16Z</responseDate>
<request verb="ListRecords" metadataPrefix="oai_dc" resumptionToken="">http://pubs.cs.uct.ac.za/perl/oai2</request>
<ListRecords><record><header>
<identifier>oai:techreports.cs.uct.ac.za:762</identifier><datestamp>2012-02-09</datestamp>
<setSpec>796561723D32303132</setSpec><setSpec>7374617475733D707562</setSpec>
<setSpec>7375626A656374733D4A:4A33</setSpec><setSpec>7375626A656374733D49:4936</setSpec>
<setSpec>7375626A656374733D4A:4A32</setSpec><setSpec>747970653D6A6F75726E616C70</setSpec>
</header><metadata>
<oai_dc:dc xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/ http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
<dc:title>Conformational properties of two exopolysaccharides produced by Inquilinus limosus, a cystic fibrosis lung
pathogen</dc:title>
<dc:creator>Kuttel, Michelle</dc:creator><dc:creator>Ravenscroft, Neil</dc:creator>
<dc:creator>Foschiatti, Michaela</dc:creator><dc:creator>Cescutti, Paola</dc:creator>
<dc:creator>Rizzo, Roberto</dc:creator>
<dc:subject>J.3 LIFE AND MEDICAL SCIENCES</dc:subject>
<dc:subject>J.2 PHYSICAL SCIENCES AND ENGINEERING</dc:subject>
<dc:subject>I.6 SIMULATION AND MODELING</dc:subject>
<dc:description>Inquilinus limosus is a multi-resistant bacterium found in the respiratory tract of patients with cystic fibrosis. This
bacterium produces two unique fully pyruvylated exopolysaccharides in similar quantities: an alpha-(1->2)-linked mannan and a
beta-(1->3)-linked glucan. We employed molecular modelling methods to probe the characteristic conformations and dynamics of
these polysaccharides, with corroboration from potentiometric titrations and circular dichroism experiments. Our calculations reveal
different structural motifs for the mannan and glucan polysaccharides: the glucan forms primarily right-handed helices with a wide
range of extensions, while the mannan forms only left-handed helices. This finding is supported by our circular dichroism
experiments. Our calculations also show that the (1->3)-beta-D-Glcp linkage is more dynamically flexible than the (1->2)-alpha-D-
Manp: the glucan characteristically forms a range of wide helices with large central cavities. In contrast, the mannan forms rigid
regular 'bottlebrush' helices with a minimal central cavity. The widely different character of these two polymers suggests a possible
differentiation of biological roles.</dc:description>
<dc:date>2012-03-01</dc:date>
<dc:type>Journal (Paginated)</dc:type>
<dc:identifier>http://pubs.cs.uct.ac.za/archive/00000762/</dc:identifier>
</oai_dc:dc></metadata></record>
<resumptionToken>100/47381799/oai_dc</resumptionToken>
</ListRecords></OAI-PMH>
```


Figure 5.16: OAI-PMH ListRecords Response

5.4 Evaluation Conclusions

Three distinct forms of evaluation were conducted; an implementation case study, a user understandability evaluation and entropy calculations.
The use case confirmed that it is possible to implement simple but still efficient protocol services.
Overall, for the understandability, the individual HIP-CF protocols were rated as being simpler to understand, having a better presentation/writing style and being easier to implement, when compared to their counterparts.
For the evaluation of the whole suite there was a 50/50 split on the ratings, with half of the participants rating HIP-CF services higher while the other half rated SRU and OAI-PMH higher, but from both groups the ratings of two protocols that offer similar services were very close to each other.
Previous knowledge of interoperability protocols played a more influential role in participants' ratings than their level of programming skills did, so much so that in some cases participants with a higher level of programming skills level rated their perceived level of programming difficulty higher than participants with less programming skill but who had previous knowledge of protocols.
The idea of a suite of protocols as opposed to the current set of individual protocols has been well received by all participants of the complete evaluation.
Simplicity has also been well received by the majority of participants, however, some participants felt that the HIP-CF suite could benefit from more functionality and less flexibility, but the general response towards HIP-CF was positive.
The entropy calculations in both cases, showed that using Shannon's entropy formula, it is more cost efficient to encode an HIP-CF message than it is to encode its counterparts, and in some cases by a very big difference of bits required.
The results above suggest that the answer to the research question: ***Is it possible to develop a uniform suite of simple and efficient interoperability protocols to improve on the current medley of protocols?*** is yes.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this research it has been hypothesised that, if interoperability is made simpler, then it increases adoption levels, making it is easier for programmers to understand and implement protocols, therefore leading to more interoperable systems.
To that end, an experimental suite of protocols was designed, implemented and evaluated. The usability evaluation tested to see how programmers would react to an alternative suite of protocols. The evidence from consulting programmers showed that they would rather implement the suite of experimental protocols than the existing set because of greater understandability which then leads to an easier to implement protocol. That evidence suggests that the suite of experimental protocols are easier to implement.
This research does not suggest that anybody should ever implement the HIP-CF protocols. Rather what it says is that there is enough evidence from the experimental study to suggest that it is possible to do better than we are currently doing, and also calls attention to the possibility of a new route for the design of high-level interoperability protocols.
The user study showed that HIP-CF could be a better alternative, which implies that the existing set of protocols are not necessarily as good as they can be.
The entropy analyses showed quantitative evidence that the HIP-CF protocol's minimalistic approach results in more efficient encoding. The case study demonstrated the feasibility of implementing and efficiently providing high-level interoperability services through the proposed simplistic suite of protocols.
This research has shown that the current set of protocols can be substantially improved on. These improvements could, and maybe should, be the result of a deeper analysis of the goals of today's protocols and also a collaboration amongst the different groups that design high-level interoperability protocols.

6.2 Future Work

If the findings from this research are to be taken into consideration and used, a number of steps would be required to be able to achieve the best possible interoperability and integration of protocol services.Some of these steps are:

- The creation of a standardised vocabulary to be used for high-level interoperability protocols. This eliminates any ambiguity in terms of understanding what a specific word/verb/request means for a particular service.
- The creation of standardised unique identifiers for digital objects. This would be particularly helpful for communities implementing more than one interoperability services, for example a user creating a collection of documents that are harvested and searched from multiple servers.
- This current set of protocols adhere to a more simplistic way of presenting their content. These are well known and trusted standards, but to keep users and welcome new ones, they could benefit if protocol developers include simplicity in the design requirements list. At an initial stage or as a test stage, communities could implement a quick reference guide to their protocol implementation. This guide would give users the minimal set of necessary rules to implement a specific protocol and, if needed, they could refer to the full documentation for a more detailed and complete analysis.
- The creation of one suite of protocols that supports multiple the high level interoperability services with multiple features support for each service.

Just as important, if not more important, than the implementation experiments mentioned above are the possibilities for further research experiments.
Possible new areas to be experimented with and explored in terms of high-level interoperability protocols research are:

- Design and evaluation of a complete experimental protocol suite that supports more interoperability services than the three used for this research, i.e. search, browse, harvest, syndication, deposit, etc.
- Evaluation matrix. The design of a matrix that includes all possible combinations of protocol evaluation techniques, for evaluating various aspects of protocol design. Such a matrix could allow designers to choose the combination of evaluation techniques that is best suited to prove/disprove the research hypotheses.
- Automated integration of simplistic suite of protocols into digital archives. Design tools that would allow archive managers to create support for a protocol by applying a few minor changes through an easy to follow set-up process.
- Automated simplification of interoperability protocols. By creating a simplification middleware between the archive and the protocol(s).

- Flexibility vs. Implementation Complexity: a study to find the balance between flexibility and implementation complexity, using HIP-CF flexible services and comparing them to a less flexible implementation. The HIP-CF services presented here combine simplicity and flexibility and as such support for some features is recommend but not mandatory. While that gives the programmers a lot of flexibility to choose which tools and methods to use it may in some cases increase complexity at the implementation level, with programmers having to cater for multiple possible options (associated with non-binding recommendations) for what can be large sets of clients and servers applications. Interesting insight of whether programmers favour flexible but complex implementations or not flexible but simpler implementations could be obtained.

  Some specific examples where less flexibility and less complexity could be applicable in the HIP-CF framework are:

  - In section 4.4.4.1.2 (Xharvester - metadataFormat) the sentence "The use of this parameter with any other request **SHOULD** generate an error" would be changed to mandatory and became "The use of this parameter with any other request **MUST** generate an error". The same would work for section 4.4.4.1.3 (Xharvester - resumptionToken) where "The use of this parameter with any other request **SHOULD** generate an error" would change to "The use of this parameter with any other request **MUST** generate an error".
  - In section 4.4.4.1.3 the option to have either an empty resumptionToken or no resumptionToken would be eliminated and only one of the options would be used as a mandatory implementation guideline.
  - In section 4.4.4.1.3 "The use of the resumptionToken after the expiryDate **SHOULD** generate an error. The dates used for expiryDate, from and until **SHOULD** follow the W3C11 Date and Time Formats for Coordinated Universal Time" would change to "The use of the resumptionToken after the expiryDate **MUST** generate an error. The dates used for expiryDate, from and until **MUST** follow the W3C11 Date and Time Formats for Coordinated Universal Time".

- Improve and extend existing protocols: This can be done but running a user study similar to the one in chapter 3. Once shortcomings for the target protocol have been identified then the appropriate measures can be taken to improve on the protocol. This can be used an extra tool for protocol designers to decide on the changes necessary for each successive version of a protocol that is released.

- Implement the HIP-CF services using different tools to the ones used for this research, for example use CQL instead of MySQL, using Atom's <link rel="next" href="www.example.com"> instead of the resumptionToken, and JSON instead of XML and assess with tools help achieve a higher level of interoperability.

Bibliography

[1] Abargues, C., Granell, C., D'az, L., and Huerta, J. Aggregating geoprocessing services using the OAI-ORE data model. *International Journal on Advances in Intelligent Systems 3* (2010), 200–210.

[2] Allinson, J., Carr, L., Downing, J., Flanders, D., Francois, S., Jones, R., Lewis, S., Morrey, M., Robson, G., and Taylor, N. *SWORD AtomPub Profile version 1.3.* AtomPub, August 2008. http://www.swordapp.org/docs/sword-profile-1.3.html Last accessed 16-08-2009.

[3] Anderson, P. *What is Web 2.0? Ideas, Technologies and Implications for Education.* JISC - Technology Standards Watch, February 2007.

[4] ANSI/NISO. *Information REtrieval (Z39.50): Application Service Definition and Protocol Specification.* The National Information Standards Organisation, 2003.

[5] Arms, W. Manuscript of digital libraries. M.I.T. Press, 2000. Last accessed on the 21-11-2011.

[6] Arms, W. Y. Key concepts in the architecture of the digital library. In *D-Lib Magazine.* D-Lib, July 1995.

[7] atompub.org. Atom publishing protocol - popularity. http://atompub.org/popularity.html, 2010.

[8] Bell, T. C., Cleary, J. G., and Witten, I. H. *Text Compression.* Prentice Hall, Inc, 1990.

[9] Berners-Lee, T., Fielding, R., and Masinter, L. *Uniform Resource Identifiers (URI): Generic Syntax.* Network Working Group, August 1998.

[10] Bradner, S. *RFC2119 Key words for use in RFCs to Indicate Requirement Levels,* March 1997. Last accessed on the 09-06-2011.

[11] Brooks, S., Donovan, P., and Rumble, C. Developing nations, the digital divide and research databases. Elsevier Inc., November 2005.

[12] Castelli, D., and Pagano, P. A system for building expandable digital libraries. In *JCDL'03 - Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries* (2033), I. C. Society, Ed., Springer, p. 335345.

[13] Chumbe, S., MacLeod, R., Barker, P., Moffat, M., and Rist, R. Overcoming the obstacles of harvesting and searching digital repositories from federated searching toolkits, and embedding them in vles. In *Conference Paper in the 2nd International Conference on Computer Science and Information Systems* (June 2006). http://hdl.handle.net/10760/7629.

[14] Cornell Digital Library Research Group. *Dienst Protocol Version 4.1 - Draft*, Februay 1994.

[15] Crocker, D. H. *RFC822 Standard for the Format of ARPA Internet Text Messages*, August 1982.

[16] Currier, S. Sword: Cutting through the red tape to populate learning materials repositories. In *SWORD*. JISC e-Learning Focus, February 2009, p. 28.

[17] Davis, J., Fielding, D., Lagoze, C., and Marisa, R. *Dienst Protocol Specifications*, May 2000.

[18] Davis, J. R., and Lagoze, C. *Dienst, A Protocol for a Distributed Digital Document Library*, August 1994.

[19] Davis, J. R., and Lagoze, C. Ncstrl: Design and deployment of a globally distributed digital library. *JASIS 51*, 3 (2000), 273–280.

[20] Dempsey, L., and Weibel, S. L. The warwick metadata workshop: A framework for thedeployment of resource description. In *D-Lib Magazine*. D-Lib, July/Agust 1996.

[21] eBizMBA The eBusiness Knowledgebase. Top 15 most popular search engines - february 2012. http://www.ebizmba.com/articles/search-engines, February 2012. Last accessed on 13-02-2012.

[22] European Commission - Information Society and Media. Driver-ii digital repositories infrastructure vision for european research - framework programme 7 (2007-2013) research infrastructures projects.

[23] Feng, D., Siu, W. C., and Zhang, H. J. *Multimedia Information Retrieval and Management: Technological Fundamentals and Applications*. Signals and Communnication Technology. Springer, 2003.

[24] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. *RFC2616 - HypertextTransfer Protocol: HTTP/1.1*, June 1999.

[25] Fielding, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. tdissertation, University of California, Irvine, 2000.

[26] Franks, J., Hallan-Baker, P., Hostetler, J., Laurence, S., Leach, P., Luotonen, A., and Steward, L., Eds. *RFC2617 - HTTP Authentication: Basic Digest Access Authentication* (June 1999), ACM.

[27] GIGLIA, E. Report on oai 6: Cern workshop on innovations in scholarly communication. geneva 17-19 june 2009. In *D-Lib Magazine*, vol. 15, Number 9/10. D-Lib Magazine, September/October, September/October 2009. Last accessed on the 21th of September 2009.

[28] GRADMANN, S. *Interoperability. A key concept for large scale, persistent digital libraries. A briefing paper from digital preservation Europe.*

[29] GREEN, N., IPEIROTIS, P. G., AND GRAVANO, L. Sdlip + starts = sdarts - a protocol and toolkit for metasearching. In *Proceedings of the 2007 Joint Conference on Digital Libraries (JCDL)* (June 2001), JCDL, ACM.

[30] GREGORIO, J., AND DE HORA, B. *RFC5023 - The Atom Publishing Protocol.* Network Working Group, October 2007.

[31] HARRISON, T. L., NELSON, M. L., AND ZUBAIR, M. The dienst-oai gateway. In *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries* (2003), pp. 309–311. http://dl.acm.org/citation.cfm?id=827140.827194.

[32] HOOGERWERF, M. Durable enhanced publications. http://www.ais.up.ac.za/digi/docs/hoogerwerf_paper.pdf. Last accessed on the 21th of September 2009.

[33] HOUSLEY, S. Rss security. http://www.feedforall.com/rss-security.htm.

[34] JISC. Jisc information environment architecture. In *Glossary.* JISC UKOLN, 2005.

[35] JISC-UKOLN. Sword app. http://swordapp.org/. Last accessed on the 16-08-2009.

[36] KAHN, R., AND WILENSKY, R. *A Framework for Distributed Digital Object Services*, May 1995.

[37] KAHN, R., AND WILENSKY, R. A framework for distributed digital object services. *International Journal on Digital Libraries 6*, 2 (2006), 115–123.

[38] KLYNE, G., AND NEWMAN, C. *RFC3339. Date and Time on the Internet: Timestamps*, July 2002.

[39] KUNZE, J., AND BAKER, T. *The Dublin Core Metadata Element Set.* Network Working Group, August 2007. www.ietf.org/rfc/rfc5013.txt.

[40] LAGOZE, C. The warwick framework - a container architecture for diverse sets of metadata. In *D-Lib Magazine.* D-Lib, july/August 1996. http://www.dlib.org/dlib/july96/lagoze/07lagoze.html.

[41] LAGOZE, C., LYNCH, C. A., AND JR., R. D. The warwick framework: A container architecture for aggregating sets of metadata. Tech. rep., Cornell University, 1996. http://hdl.handle.net/1813/7248.

[42] LAGOZE, C., AND VAN DE SOMPEL, H. The open archives initiative: building a low-barrier interoperability framework. In *Proceedings of the 1st ACM/IEEE-CS joint conference on Digital Libraries (Roanoke, VA, USA)* (2001), ACM Publishers.

[43] LAGOZE, C., VAN DE SOMPEL, H., JOHNSTON, P., NELSON, M., SANDERSON, R., AND WARNER, S. *Open Archives Initiative Object Reuse and Exchange. Ore Specification - Abstract Data Model.* Open Archives Initiative, October 2008.

[44] LAGOZE, C., VAN DE SOMPEL, H., NELSON, M., AND WARNER, S. *Open Archives Initiative Frequently Asked Questions (FAQ).* Last accessed on the 21st of May 2009.

[45] LAGOZE, C., VAN DE SOMPEL, H., NELSON, M., AND WARNER, S. *The Open Archives Initiative Protocol for Metadata Harvesting.* The Open Archives Initiative, http://www.openarchives.org/OAI/2.0/openarchivesprotocol.html, December 2008. Last accessed on the 13th of September 2009.

[46] LIBRARY OF CONGRESS. *LC Z39.50/SRW/SRU Server Configuration Guidelines.* http://www.loc.gov/z3950/lcserver.html.

[47] LIBRARY OF CONGRESS. *CQL: COntextual Query Language (SRU VErsion 1.2 Specifications)*, 2008. http://www.loc.gov/standards/sru/specs/cql.html.

[48] LIBRARY OF CONGRESS. *SRU Protocol Transport (SRU Version 1.2 Specifications)*, February 2008. http://www.loc.gov/standards/sru/specs/transport.html.

[49] LYNCH, C. *RFC1729 - Using the Z39.50 Information Retrieval Protocol.* Network Working Group, December 1994.

[50] LYNCH, C. The z39.50 information retrieval standard. part 1: A strategic view of its past, present and future. *D-Lib Magazine* (April 1997).

[51] MANOLA, F., AND MILLER, E. *REF Primer.* W3C, February 2004. Last accessed on the 06-08-2009.

[52] MILLER, P. Interoperability. what is it and why should i want it? In *Ariadne.* Ariadne24, June 2000.

[53] MORADI, H., GRZYMALA-BUSSE, J. W., AND ROBERTS, J. A. Entropy of english text: Experiments with humans and a machine learning system based on rough sets. *An International Journal of Information Sciences 104* (1998), 31–47.

[54] MORGAN, E. L. An introduction to the search/retrieve url service (sru). *Ariadne*, Issue 40 (July 2004). http://www.ariadne.ac.uk/issue40/morgan.

[55] MULLER, U., AND J. WANG. Open archives forum: Evaluative review of metadata harvesting pilot. In *Open Archives Forum.* Open Archives Initiative, September 2003.

[56] NELSON, M., AND VAN DE SOMPEL, H. Ijdl special issue on complex digital objects: Guest editors' introduction. *International Journal on Digital Libraries 6*, 2 (2006), 113–114.

[57] NELSON, M., VAN DE SOMPEL, H., AND WARNER, S. Advanced overview of versions 2.0 of the open archives initiative for metadata harvesting. http:www.cs.odu.edu/ mln/jcdl03/, May 2003.

[58] NEWBY, G. B., AND FRANKS, C. Distributed proofreading. In *Proceedings of the 2003 Joint Conference on Digital Libraries (JCDL)* (2003).

[59] NOTTINGHAM, M., AND SAYRE, P. *RFC4287 - The Atom Syndication Format*, December 2005.

[60] OAIFORUM. *The Open Archives Forum Online Tutorial.* Last accessed on the 22-05-2009.

[61] OPEN DLIB. Opendlib - digital library service system. http://www.opendlib.com/area4/info_arch05.html.

[62] OPENSEARCH.ORG. Opensearch. www.opensearch.org/Home. Last accessed on the 26-05-2012.

[63] PAEPCKE, A., BRANDRIFF, R., JANEE, G., LARSON, R., LUDAESCHER, B., MELNIK, S., AND RAGHAVAN, S. Search middleware and the simple digital library interoperability protocol. In *D-Lib Magazine*. D-Lib, March 2000. http://www.dlib.org/dlib/march00/paepcke/03paepcke.html.

[64] PAISLEY, W. J. The effects of authorship, topic, structure, and time of composition on letter redundancy in english texts. *JOURNAL OF VERBAL LEARNING AND VERBAL BEHAVIOR 5* (1966).

[65] PAPAZOGLOU, M. *Web Services: Principles and Technology*, 1 ed. Pearson Education Limited, 2008.

[66] PARZIALE, L., BRITT, D. T., DAVIS, C., FORRESTER, J., LIU, W., MATTHEWS, C., AND ROSSELOT, N. *TCP/IP Tutorial and Technical Overview*, 8th edition ed. RedBooks. IBM, December 2006.

[67] PASKIN, N. Identifier interoperability, a report on two recent iso activities. In *D-Lib Magazine*, vol. 12. D-Lib Magazine, April 2006. www.dlib.org/dlib/april06/paskin/04paskin.html.

[68] POUNTAIN, D. *Concise Dictionary of Computing.* Penguin Books, 2003.

[69] PROJECT GUTENBERG. Free ebooks by project gutenberg. http://www.gutenberg.org. Last accessed on 11-02-2012.

[70] PROJECT GUTENBERG. Project gutenberg news. http://www.gutenbergnews.org/about/. Last accessed on 11-02-2012.

[71] RESCORLA, E. *RFC2818 - HTTP over TLS*, May 2000.

[72] RSS ADVISORY BOARD. Really simple syndication specifications. tutorials and discussion. specification history. http://www.rssboard.org/rss-history. Last accessed on the 02-08-2009.

[73] RSS ADVISORY BOARD. Rss autodiscovery. http://www.rssboard.org/rss-autodiscovery, November 2006.

[74] RSS ADVISORY BOARD. Really simple syndication specifications, tutorials and discussion. rss 2.0 specifications, version 2.0.11. http://www.rssboard.org/rss-specification, March 2009. Last accessed om the 02-08-2009.

[75] RSS ADVISORY BOARD. Rss specification. http://www.rssboard.org/rss-specification, March 2009.

[76] RUBY, S. Rss 2.0 and atom 1.0 compared. http://www.intertwingly.net/wiki/pie/Rss20AndAtom10Compared, September 2008.

[77] SHANNON, C. E. Prediction and entropy of printed english. Bell System Technical Journal, September 1950.

[78] SMOTHERS, V. Knowing when to rest: Simple object access protocol vs. representational state transfer web services. version 1.0, June 2009. Last accessed on the 05-10-2009.

[79] SNELL, J. Getting to know the atom publishing protocol, part 1: Create and edit web resources with the atom publishing protocol. In *IBM*. IBM, October 2006. Last accessed on the 06-04-2009.

[80] SNELL, J. Getting to know the atom publishing protocol, part 2: Put the atom publishing protocol (app) to work. http://www.ibm.com/developerworks/library/x-atompp2/, November 2006. Last accessed on 06-02-2012.

[81] SNELL, J. *Atom License Extension (RFC 4946)*, July 2007. http://www.ietf.org/rfc/rfc4946.txt.

[82] SOFTWARE GARDEN. What is rss? http://rss.softwaregarden.com/aboutrss.html, July 2004. Last accessed on the 06-04-2009.

[83] SRW EDITORIAL BOARD. Srw: Search/retrieve webservice. http://srw.cheshire3.org/SRW-1.1.pdf, May 2004.

[84] STANFORD DIGITAL LIBRARIES TECHNOLOGIES. *The Simple Digital Library Interoperability Protocol (SDLIP-Asynch)*. Last accessed on the 08-02-2012.

[85] STANFORD DIGITAL LIBRARIES TECHNOLOGIES. *The Simple Digital Library Interoperability Protocol (SDLIP-Core)*. Last accessed on the 08-02-2012.

[86] Stewart, G., Crane, G., and Babeu, A. A new generation of textual corpora - mining corpora from very large collections. In *Proceedings of the 2007 Joint Conference on Digital Libraries (JCDL)* (June 2007), JCDL, ACM.

[87] Suleman, H. *Open Digital Libraries.* PhD thesis, Virginia Polytechnic Institute and State University, November 2002.

[88] Suleman, H., and Fox, E. A. A framework for building open digital libraries. In *D-Lib Magazine*, vol. 7. D-Lib, December 2001. http://www.dlib.org/dlib/december01/suleman/12suleman.html.

[89] Suleman, H., and Fox, E. A. Designing protocols in support of digital library componentization. In *Lecture Notes in Computer Science* (2002), vol. 2539, Springer, pp. 217–226.

[90] Surf Foundation. Emerging standards for enhanced publications and repository technology. In *Survey on Technology.* Amsterdam University Press, 2009.

[91] Van de Sompel, H., Krichel, T., Nelson, M., Hochstenbach, P., Lyapunov, V., Maly, K., Zubair, M., Kholief, M., Liu, X., and and, H. O. The ups prototype: An experimental end-user service across e-print archives. *D-Lib Magazine Volume 6*, Number 2 (February 2000). http://www.dlib.org/dlib/february00/vandesompel-ups/02vandesompel-ups.html.

[92] Van de Sompel, H., and Lagoze, C. The santa fe convention of the open archives initiative. *D-Lib Magazine Volume 6*, Number 2 (February 2000). http://www.dlib.org/dlib/february00/vandesompel-oai/02vandesompel-oai.html.

[93] W3Schools. *Platform for Internet Content Selection (PICS).* W3C. http://www.w3.org/PICS/.

[94] W3Schools. Xml information set (second edition). http://www.w3.org/TR/2004/REC-xml-infoset-20040204/, February 2004. Last accessed on the 21st of September 2009.

[95] W3Schools. Extensible markup language (xml) 1.0. http://www.w3.org/TR/REC-xml/, November 2008. Last accessed on the 02-08-2009.

[96] Wall, A. History of search engines: From 1945 to google today. http://www.searchenginehistory.com/.

[97] Wikipedia. Opensearch. en.wikipedia.org/wiki/OpenSearch. Last accessed on the 26-05-2012.

[98] Wikipedia. Rss. en.wikipedia.org/wiki/RSS.

[99] Witten, I., Boddie, S., Bainbridge, D., and MacNab, R. Greenstone: a comprehensive open-source digital library system. In *Proceedings of the Fifth ACM Conference on Digital Libraries* (2000), ACM, pp. 113–121.

[100] WITTENBRINK, H. *RSS And Atom: Understanding And Implementing Content Feeds And Syndication.* Packt Pulishing, November 2005.

[101] WU, W., AND LI, J. Rss made easy: A basic guide for librarians. *Medical Reference Quarterly Vol.26(1)* (2007).

Appendices

Appendix A: RSS Channel Elements

Element	Occurrence	Description
title	Mandatory	The name of the channel. It should match the title of a website containing the same information as the RSS file.
link	Mandatory	The URL to the website corresponding to the channel.
description	Mandatory	A sentence that describes the content of the channel.
language	Optional	The language in which the channel is written.
copyright	Optional	A notice of the copyrights for the content in the channel.
managingEditor	Optional	The email address of the editor.
webmaster	Optional	The email address of the Web administrator.
pubDate	Optional	The date at which the content was published.
lastBuildDate	Optional	The last time the content was edited.
category	Optional	Specifies one or more categories that the channel belongs to.
generator	Optional	A string indicating the program used to generate the channel.
docs	Optional	The URL that points to the documentation for the format used in the RSS file.
cloud	Optional	Implements a lightweight publish-subscribe protocol for RSS feeds that allows processes to register with a cloud and be notified of updates to the channel.
ttl	Optional	Time to live or ttl indicates the time (in minutes) a channel can be cached before refreshing from the source.
image	Optional	Specifies a GIF, JPEG or PNG image that can be displayed with the channel.
rating	Optional	the PICS (Platform for Internet Content Selection) [93], rating for the channel.
textInput	Optional	Specifies a text input box that can be displayed with the channel.
skipHours	Optional	Indicates the hours during which aggregators may not read the channel.
skipDays	Optional	Indicates the days during which aggregators do not need to read the channel.

Appendix B: Atom Elements

Element	Type	Description
atom:feed	Container	The top level element of an Atom feed document. It acts as a container for the data and metadata associated with the feed.
atom:entry	Container	Represents an individual entry. It acts as a container for data and metadata associated with that specific entry. The *atom:entry* element can appear as a child of the *atom:feed* element or it can appear as a stand-alone Atom Entry document.
atom:content	Container	A language-sensitive element that contains either content or links to the content of an entry.
atom:author	Metadata	A Person construct that indicates the author of the entry or feed.
atom:category	Metadata	Indicates the category associated with an entry or feed.
atom:contributer	Metadata	A Person construct that indicates a person or an entity who contributed to the feed or entry.
atom:generator	Metadata	Identifies the agent used to generate a feed. This information is used for debugging and other purposes.
atom:icon	Metadata	Contains an URI that contains an image that shows the visual representation of a feed.
atom:id	Metadata	Presents the unique identifier of an entry or feed.
atom:link	Metadata	Defines a reference that links an entry or feed to a Web resource.
atom:published	Metadata	A Date construct that indicates the time of an event early in the life-cycle of the entry .
atom:rights	Metadata	A Text construct that provides information about who or what institution holds the rights over an entry or feed .
atom:source	Metadata	Stores the atom:feed metadata for an atom:entry that is copied from one feed to another.
atom:subtitle	Metadata	A Text construct that shows the description or sub- title of a feed.
atom:summary	Metadata	A Text construct that contains an excerpt of an entry.
atom:title	Metadata	A Text construct that contains the title of an entry or feed.
atom:updated	Metadata	A Date construct that indicates when last an entry or feed was modified in a way the publisher considers significant.

Appendix C: User Online Survey Questionnaire

Survey Questionnaire

1. **What is your level of confidence with each of the following protocols? Choose your answer from 1 to 5 according to the values explained below.**
Choose your answer according to the following values:
1 - Expert implementer
2 - Implemented (have written code for an implementation of the protocol)
3 - Read and understood
4 - Heard about it, but do not know the details
5 - Never heard about it

Please choose the appropriate response for each item:

RSS	1☐	2☐	3☐	4☐	5☐
ATOM	1☐	2☐	3☐	4☐	5☐
APP	1☐	2☐	3☐	4☐	5☐
Z39.50	1☐	2☐	3☐	4☐	5☐
OAI-PMH	1☐	2☐	3☐	4☐	5☐
OAI-ORE	1☐	2☐	3☐	4☐	5☐
SRU/W	1☐	2☐	3☐	4☐	5☐
SWORD	1☐	2☐	3☐	4☐	5☐
Other(s)	1☐	2☐	3☐	4☐	5☐

2. **If in question one you choose other(s), please name the other protocol/s that you are familiar with.** Write the name of the protocol followed by the number that indicates your level of confidence with the protocol. If there is more than one protocol separate the list with commas (,).

---

3. **What in your opinion is the most useful feature(s) of each of the protocols?**

RSS: ☐
ATOM: ☐
APP: ☐
Z39.50: ☐
OAI-PMH: ☐
OAI-ORE: ☐
SRU/W: ☐
SWORD: ☐
Other(s): ☐

4. **What in your opinion is the least useful feature(s) of each of the protocol?**

RSS:
ATOM:
APP:
Z39.50:
OAI-PMH:
OAI-ORE:
SRU/W:
SWORD:
Other(s):

5. **How do you believe these protocols could be improved?**

Your opinion on improvements for interoperability protocols in general. If you are not making a generalised comment please name the protocol before making the actual comment.

6. **General comments about the protocols. If you are not making a general comment please name the protocol before making the actual comment.**

Appendix D: User Evaluation Consent Form

Meta-standardisation of Interoperability Protocols
Consent Form

I....., hereby agree to take part in this experiment/ evaluation session. I understand that I will be asked to read documentations of interoperability protocols and then given my understanding, I will answer a questionnaire which will be used to collect data for a Masters research project evaluation. I understand that the aim of this experiment is to collect data to compare different protocols that provide the same service, and that the results may potentially bring changes to currently used interoperability protocols. I understand that when reporting on results the researcher will not use my personal information. I agree to forgo of any rights that may arise from the data I provide or the research results of the project.

Signature.....
Date.....

Appendix E: Understandability Complexity Questionnaire

E1: Individual Protocols Evaluation Questionnaire

Thanks for participating in this study. This is part of the evaluation of a Masters project called Meta-standardisation of Interoperability Protocols.

Task:
The aim of this study is to assess which protocol service is easier to understand and seems simpler to implement.
You have been given the documentation of two distinct protocols to read. Based on your understanding of each protocol please answer the questions below.
Please tick the box corresponding to your academic level.

Undergraduate Degree Student/Holder: ☐
Honours Degree Student/Holder: ☐
Masters Degree Student/Holder : ☐
Ph.D. Student/Holder or Higher: ☐

Question 1
Name the two protocols for which you were given the documentation to read?
Protocol 1: _____
Protocol 2: _____

Question 2
How confident are you as a programmer? Specify your highest level of confidence in the programming language(s) you use.

1. I'm an expert programmer ☐
2. I'm a good programmer ☐
3. I'm an average programmer ☐
4. I'm a beginner ☐
5. I'm not a programmer ☐

Question 3
Before this evaluation session where you familiar with these or any other high-level interoperability protocols?
Yes ☐ **No** ☐

Question 4
If your answer to question 3 was yes, please specify which protocol(s) and your level of expertise based on the scale below (Example: Protocol Name - 2) :
1. Expert 2. Average User 3. Understand 4. know about it

Protocol: _____
Protocol: _____
Protocol: _____
Protocol: _____
Protocol: _____

Note: For the questions below protocol 1 and protocol 2 refer to the protocols as assigned in question 1. When the same answer applies to a question on both sides (e.g. question 5 has the same answer for both protocol 1 and protocol 2), you may answer just on one side and indicate the answer applies to both protocols by writing "BOTH" at the end of the question.

PROTOCOL 1_____	PROTOCOL 2_____
Question 3 What kind of service does protocol 1 provide?_____ _____ _____	**Question 3** What kind of service does protocol 2 provide?_____ _____ _____
Question 6 On a scale of 1 - 10 , 1 being **I still don't understand the protocol** and 10 being **I now have a good understanding of the protocol**, rate your understanding of protocol 1? 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) 8( ) 9( ) 10( )	**Question 6** On a scale of 1 - 10 , 1 being **I still don't understand the protocol** and 10 being **I now have a good understanding of the protocol**, rate your understanding of protocol 2? 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) 8( ) 9( ) 10( )
Question 7 On a scale of 1 - 10 , 1 being **poor** and 10 being**very good**, rate the writing /presentation style of protocol 1. 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) 8( ) 9( ) 10( )	**Question 7** On a scale of 1 - 10 , 1 being **poor** and 10 being**very good**, rate the writing /presentation style of protocol 2. 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) 8( ) 9( ) 10( )
Question 8 On a scale of 1 - 10 , 1 being **easy** and 10 being **difficult**, rate your idea of what an implementation of protocol 1 would be. 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) 8( ) 9( ) 10( )	**Question 8** On a scale of 1 - 10 , 1 being **easy** and 10 being **difficult**, rate your idea of what an implementation of protocol 2 would be. 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) 8( ) 9( ) 10( )
Question 9 General comments about protocol 1 (e.g. critiques , suggestions, tips for improvements, notes). _____ _____ _____ _____ _____ _____ _____ _____ _____ _____ _____ _____	**Question 9** General comments about protocol 1 (e.g. critiques , suggestions, tips for improvements, notes). _____ _____ _____ _____ _____ _____ _____ _____ _____ _____ _____ _____

E2: Suite of Protocols Evaluation Questionnaire

Thanks for participating in this study. This is part of the evaluation of a Masters project called Meta-standardisation of Interoperability Protocols.

Task:
The aim of this study is to assess which protocol service is easier to understand and seems simpler to implement. And also assess the feasibility having of a suite of protocol services.
You have been given the documentation of three distinct protocols to read. Based on your understanding of each protocol please answer the questions below.
Please tick the box corresponding to your academic level.

Undergraduate Degree Student/Holder: ☐
Honours Degree Student/Holder: ☐
Masters Degree Student/Holder : ☐
Ph.D. Student/Holder or Higher: ☐

Question 1
Name the protocol you were given the documentation to read?
Protocol 1: _____
Protocol 2: _____
Protocol 3: _____

Question 2
How confident are you as a programmer? Specify your highest level of confidence in the programming language(s) you use.

1. I'm an expert programmer ☐
2. I'm a good programmer ☐
3. I'm an average programmer ☐
4. I'm a beginner ☐
5. I'm not a programmer ☐

Question 3
Before this evaluation session where you familiar with these or any other high-level interoperability protocols?
Yes ☐ **No** ☐

Question 4
If your answer to question 3 was yes, please specify which protocol(s) and your level of expertise based on the scale below (Example: Protocol Name - 2) :
1. Expert 2. Average User 3. Understand 4. know about it

Protocol: _____
Protocol: _____

Protocol: _____
Protocol: _____
Protocol: _____

Note: For the questions below protocols 1, 2 and 3 refer to the protocols as assigned in question 1.

Question 5 What kind of service(s) does protocol 1 provides/supports? _____ _____ _____ _____ _____	**Question 5** What kind of service(s) does protocol 1 provides/supports? _____ _____ _____ _____ _____	**Question 5** What kind of service(s) does protocol 1 provides/supports? _____ _____ _____ _____ _____
Question 6 On a scale of 1 - 10 , 1 being **I still don't understand the protocol** and 10 being **I now have a good understanding of the protocol**, rate your understanding of protocol 1? 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) 8( ) 9( ) 10( )	**Question 6** On a scale of 1 - 10 , 1 being **I still don't understand the protocol** and 10 being **I now have a good understanding of the protocol**, rate your understanding of protocol 2? 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) 8( ) 9( ) 10( )	**Question 6** On a scale of 1 - 10 , 1 being **I still don't understand the protocol** and 10 being **I now have a good understanding of the protocol**, rate your understanding of protocol 3? 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) 8( ) 9( ) 10( )
Question 7 On a scale of 1 - 10 , 1 being **poor** and 10 being **very good**, rate the writing/presentation style of protocol 1. 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) 8( ) 9( ) 10( )	**Question 7** On a scale of 1 - 10 , 1 being **poor** and 10 being **very good**, rate the writing/presentation style of protocol 2. 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) 8( ) 9( ) 10( )	**Question 7** On a scale of 1 - 10 , 1 being **poor** and 10 being **very good**, rate the writing/presentation style of protocol 3. 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) 8( ) 9( ) 10( )
Question 8 On a scale of 1 - 10 , 1 being **easy** and 10 being **difficult**, rate your idea of what an implementation of protocol 1 service(s) would be. 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) 8( ) 9( ) 10( )	**Question 8** On a scale of 1 - 10 , 1 being **easy** and 10 being **difficult**, rate your idea of what an implementation of protocol 2 service(s) would be. 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) 8( ) 9( ) 10( )	**Question 8** On a scale of 1 - 10 , 1 being **easy** and 10 being **difficult**, rate your idea of what an implementation of protocol 3 service(s) would be. 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) 8( ) 9( ) 10( )
Question 9 _____ _____ _____ _____ _____ _____ _____ _____ _____	**Question 9** _____ _____ _____ _____ _____ _____ _____ _____ _____	**Question 9** _____ _____ _____ _____ _____ _____ _____ _____ _____

Question 10

In your opinion, having a protocol that supports multiple high-level interoperability services is:

a) A great idea, since developers who implement one of the services (e.g. search) are likely to also implement the other service(s) (e.g. browsing and/or harvesting), and knowledge of the common framework may facilitate the overall process. ☐

b) A good idea, one option for multiple requirements. ☐

c) Unnecessary. The current situation works just fine. ☐

d) Bad idea, why mix the different services. It is simpler if each service is covered by an individual protocol. ☐

e) Will not work, in trying to cover too many areas the suite would end up not covering any of them properly. ☐

f) Nome of the above. ☐

If you chose option f, please elaborate on your choice._____